SECURE AND PRIVATE ONLINE COLLABORATION

by Keith Frikken

Center for Education and Research in
Information Assurance and Security,
Purdue University, West Lafayette, IN 47907-2086

SECURE AND PRIVATE ONLINE COLLABORATION

A Thesis

Submitted to the Faculty

of

Purdue University

by

Keith B. Frikken

In Partial Fulfillment of the

Requirements for the Degree

of

Doctor of Philosophy

August 2005

ACKNOWLEDGMENTS

During my five years at Purdue, I have had the opportunity to work with many great people. The person that has influenced my research the most while at Purdue, is my major professor Mikhail Atallah. I am very thankful for his advice on research and teaching.

I have had the opportunity to collaborate with many people, some of who have helped contribute ideas to this thesis including: Michael Goodrich, Roberto Tammasia, and Jiangtao Li. However, there have been many others that I have collaborated with on other work including: Marina Blanton, Sunil Prabhakar, Rei Safavi-Naini, Vinayak Deshpande, and Leroy Schwarz.

I would also like to thank many others that have contributed to my academic career while at Purdue. First, I would like to thank my committee. Also I would like to thank Susanne Hambrusch who was my GAANN fellowship advisor, and I would like to thank Greg Frederickson for serving as my teaching mentor. I would also like to thank the staff at CERIAS. Finally, I would like to thank the anonymous reviewers of the world who have made much of the work in this thesis stronger with their helpful comments.

Finally I would like to thank friends and family, especially my fiancee Sara.

Portions of this work were supported by:

Grants EIA-9903545, IIS-0219560, IIS-0312357, and IIS-0242421 from the National Science Foundation, Contract N00014-02-1-0364 from the Office of Naval Research, Intel, by sponsors of the Center for Education and Research in Information Assurance and Security, and by Purdue Discovery Park's e-enterprise Center.

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Frikken, Keith B. Ph.D., Purdue University, August, 2005. Secure and Private Online Collaboration. Major Professor: Mikhail J. Atallah.

In this thesis, we investigate confidentiality- and privacy-preserving protocols. Confidentiality- and privacy-preserving protocols (also called secure protocols) allow two or more parties to compute some function of their private inputs without revealing to any group of the parties information other than the cooperatively computed output and what can be deduced from this output and the group's individual inputs (which is unavoidable, as it is inherently part of any such protocol). It has been shown previously that any function can be computed in such a manner; the study of computing any function securely is called Secure Multiparty Computation (SMC) or Secure Function Evaluation (SFE). However, before applying these techniques to a specific domain, one first has to identify problems where secure protocols are useful. When there is a situation where a secure protocol is needed, there will always be a secure protocol for computing the function by the general results described in the SMC literature. However, the general solutions are complex and in many cases there are more efficient domain-specific solutions. In this thesis we look at several application domains including: trust negotiation, credit checking, services for location-aware devices, contract negotiation, and secure biometric authentication. In these domains we identify situations where secure protocols are useful and then develop simple and efficient protocols for these situations.

1 INTRODUCTION

Many trends in computing systems such as pervasive/ubiquitous computing lead to the availability of massive amounts of new information. The availability of this new information leads to many opportunities for collaboration, in ways that were not previously possible. However, these new collaborative situations often violate an organization's confidentiality or an individual's privacy. In such situations one of two things typically happens: The new collaboration does not occur, or more disturbingly, the collaboration occurs with careless disregard for the confidentiality and privacy concerns. The latter case is exacerbated by the fact that the individual whose privacy is being compromised is often not involved in the collaboration. However, not performing the collaboration is also undesirable, because one or more parties stand to benefit from the collaboration. Thus the question becomes whether it is possible to have the benefit of collaboration without requiring a loss of privacy, or in other words "can we have our cake and eat it too". Collaborating in a secure manner has benefits even when the parties with privacy and confidentiality concerns trust the collaborating parties. While the parties may trust each other, there is always the problem of an outsider gaining unauthorized access to some of their systems, from a malicious insider, or from spyware and other malware. Performing the computations in a secure manner makes these attacks less effective, because the sensitive information is not revealed to the other parties during the computation. In this thesis, we examine several domains and introduce confidentiality and privacy-preserving protocols for many problems and answer the above-mentioned question in the affirmative.

Confidentiality and privacy-preserving protocols allow two or more entities to compute some function on private inputs without revealing information to any group of the parties other than the cooperatively computed output and what can be deduced from this output and the group's individual inputs (which is unavoidable, as it is inherently part of any such protocol). It has been shown previously that any function can be computed in such a manner; the study of computing any function securely is called Secure Multiparty Computation (SMC) or Secure Function Evaluation (SFE). However, before applying these techniques to a specific domain, one first has to determine if a private protocol solves some privacy or confidentiality issue (e.g., if the final result still leaks private data then a secure protocol may not be useful). When a specific protocol is appropriate, there will always be a secure protocol for computing the function by the general results described in the SMC literature. However, the general solutions are complex and in many cases there are more efficient domain-specific solutions. In this thesis we look at several application domains including: trust negotiation, credit checking, services for location-aware devices, contract negotiation, and secure biometric authentication. In these domains we identify situations where secure protocols are useful and then develop simple and efficient protocols for these situations.

The specific problems we studied were selected because each of them is a good prototype of an important class of interactions that are similar to it. For example, the trust negotiation work and the credit checking work are representative of a class of interactions where off-line third-party certification of the inputs is needed, i.e., without online involvement of the third party in every certified-inputs transaction. As another example, the contract negotiations work captures the essence of the difficulty of the situations where multiple outputs are possible (only one of which must be produced) and the questions of fairness and efficiency that this raises. The problem captures a very general form of online negotiation and agreement. For example, in electro-mechanical product co-design, one "contract" clause could relate to a party's tolerance requirements for manufacturing a particular part, and to another party's ability to provide the precision-machining of that part (fully revealing one's technical prowess, and the pricing thereof, unnecessarily helps the competition). A more frivolous example is an online pre-screening interaction for a college roommate arrangement, to help decide whether there is enough compatibility for a face to face meeting (one may not wish the pre-screening interaction to reveal to a stranger one's allergies, work habits and hours, attitude towards parties, etc).

Therefore, although the problems we considered have important similarities (similar privacy-preservation framework, related solution techniques), they differ enough to cover many important facets of online collaborations.

1.1 Contributions

We have been the first to study several of the domains considered, and we have identified many interesting problems in these areas; we have also investigated previously studied domains and have expanded upon the previous work. We have introduced protocols for many problems that are more efficient than the corresponding protocols when using general solutions for SMC, and in many cases our protocols are simpler. We predict that many of the techniques introduced in this thesis will be useful in other domains as well.

1.2 Thesis Statement

There are many application domains where collaboration leads to mutual benefit, but where privacy and confidentiality concerns prevent such fruitful collaborations. In many of these domains, there are confidentiality or privacy preserving solutions that mitigate these concerns, while obtaining the benefits of collaboration. Furthermore, for many protocols this can be done in a manner that is more efficient and simple than general constructions proposed in the SMC literature.

1.3 Summary of Results

In this Section, we introduce the domains that are studied in this thesis. For each of these domains, we describe the privacy and confidentiality issues that arise, and we briefly describe the specific problems that we address. We also summarize our results for these specific problems. The full details are given in later Sections of this document.

1.3.1 Trust Negotiation

In an open environment such as the Internet, the decision to collaborate with a stranger (e.g., by granting access to a resource) is often based on the characteristics (rather than the identity) of the requester, via digital credentials: Access is granted if Alice's credentials satisfy Bob's access policy. The literature contains many scenarios in which it is desirable to carry out such trust negotiations in a privacy-preserving manner, i.e., so as to minimize the disclosure of credentials and/or of access policies. Elegant solutions have been proposed for achieving various degrees of privacy-preservation through minimal disclosure. We introduce protocols that protect both sensitive credentials and sensitive policies. That is, Alice gets the resource only if she satisfies the policy, Bob does not learn anything about Alice's credentials (not even whether Alice got access or not), and Alice learns neither Bob's policy structure nor which credentials caused her to gain access.

1.3.2 Credit Checking

Typically, when a borrower (Bob) wishes to establish a tradeline (e.g., a mortgage, an automobile loan, or a credit card) with a lender (Linda), Bob is subjected to a credit check by Linda. The credit check is done by having Linda obtain financial information about Bob in the form of a credit report. Credit reports are maintained by Credit Report Agencies, and contain a large amount of private information about individuals. To make matters more complicated, Linda's criteria for loan qualification are confidential. We propose a "privacy-preserving" credit check scheme that allows Bob to have his credit checked without divulging private information to Linda while protecting Linda's interests. That is we introduce protocols for credit checking that: i) protect Bob's private information, ii) make sure that Bob cannot lie about his credit (thus Linda is assured that the information is accurate), iii) protect Linda's qualification criteria, iv) prevent the CRA from learning anything other than "Bob requested a loan from Linda", and v) has a communication architecture that is similar to what is done in non-private credit checking today.

1.3.3 Services for Location-Aware Devices

The number of location-aware mobile devices has been rising for several years. As this trend continues, these devices may be able to use their location information to provide interesting applications for their owners. Possible applications for such devices include: determining if a specific route is near an item of interest or finding the nearest point of interest for a route. The difficulty with such computations is that the owners of the devices will not want their devices to send their location (or future locations) to a server to compute the desired function. We explore computing distance functions for routes in a private manner. We introduce secure protocols for three distance problems: i) the distance between a point and a line segment, ii) the distance between two moving points each defined by a parametric equation (with constant velocity), and iii) the distance between two line segments.

We also explore various "self-protecting" privacy solutions. Our focus is on perturbation based techniques for location based queries. In these schemes, the client will obfuscate his current location and send this query to the server. The server will then respond to this obfuscated query. Of course such a solution leads to inaccuracies in the answer of the query, but this is a tradeoff that the client can choose. Clearly, from a privacy standpoint privacy-preserving protocols are preferred to these perturbation based approaches as they reveal less information and do not introduce inaccuracy into the answer; however due to practicality reasons this error-based paradigm is useful. This error-based paradigm is particularly useful since: i) the devices do not need to use expensive cryptography and ii) the server does not need to use special software (i.e., these techniques can be used by devices when the server is not cooperative).

1.3.4 Contract Negotiation

Suppose Alice and Bob are two entities (e.g. agents, organizations, etc.) that wish to negotiate a contract. A contract consists of several clauses, and each party has certain constraints on the acceptability and desirability (i.e., a private "utility" function) of each clause. If Bob were to reveal his constraints to Alice in order to find an agreement, then she would learn an unacceptable amount of information about his business operations or strategy. To alleviate this problem we propose using a secure protocol to find an agreement between the two parties. There are two components to this: i) determining whether an agreement is possible (if not then no other information should be revealed), and ii) in case an agreement is possible, computing a contract that is *valid* (acceptable to both parties), *fair* (when many valid and good outcomes are possible one of them is selected randomly with a uniform distribution, without either party being able to control the outcome), and *efficient* (no clause is replaceable by another that is better for both parties). It is the fairness constraint in (ii) that is the centerpiece of our work in this domain.

1.3.5 Biometric Authentication

We present computationally "lightweight" schemes for performing biometric authentication that carry out the comparison stage without revealing any information that can later be used to impersonate the user (or reveal personal biometric information). Unlike some previous computationally expensive schemes—which make use of the slower cryptographic primitives—we present methods that are particularly suited to financial institutions that authenticate users with biometric smartcards, sensors, and other computationally limited devices. In our schemes, the client and server need only perform cryptographic hash computations on the feature vectors, and do not perform any expensive digital signatures or public-key encryption operations. In fact, the schemes we present have properties that make them appealing even in a framework of powerful devices capable of public-key signatures and encryptions. Our schemes make it computationally infeasible for an attacker to impersonate a user even if the attacker completely compromises the information stored at the server, including all of the server's secret keys. Likewise, our schemes make it computationally infeasible for an attacker to impersonate a user even if the attacker completely compromises the information stored at the client device (but not the biometric itself, which is assumed to remain attached to the user and is not stored on the client device in any form).

1.4 Organization of the Thesis

The rest of this work is organized as follows. In Chapter 2 we begin by formally defining secure protocols and giving a literature review. Chapter 3 contains building blocks that are used by many of our protocols; note that some of these building blocks are novel contributions to this thesis. In Chapters 4-7 we introduce protocols in the privacy-preserving framework for trust negotiation, credit checking, services for location-aware devices, contract negotiation, and biometric authentication. We summarize our contributions and describe directions for future work in Chapter 8.

2 PRELIMINARIES

In this Chapter, we give an overview of the work related to secure protocols. The purposes of this Chapter include: i) to give a formal definition of what is meant by a secure protocol, ii) to review the literature on general secure computation, and iii) to discuss several domain-specific secure protocols. For other surveys on secure protocols, see [1–3].

2.1 Overview of Secure Computation

Formal definitions of secure protocols were originally formulated by Micali and Rogaway [4] and by Beaver [5]. In this Section, we give an informal definition of secure protocols, which we formalize in Section 2.3. Suppose Alice has private input a, that Bob has private input b, and that Alice and Bob would like to engage in a protocol to compute $f(a, b)$ for some function f without revealing anything else about their private inputs. One trivial way to securely compute such a function is for Alice and Bob to respectively send a and b to some trusted third party (TTP). The TTP would then compute $f(a, b)$ and send the result to Alice and Bob. Clearly, this securely computes the function f. Unfortunately, a third party that both Alice and Bob fully trust often does not exist.

While it is not practical to use a TTP, it does give us an ideal model against which to compare the security of our protocols. Our goal will be to emulate the existence of a third party. That is, if Alice (or Bob) can compute something in the protocol, then they would have to be able to compute the same information in the trusted third party protocol. In other words, Alice (Bob) does not learn anything that cannot be deduced from a (b) and $f(a, b)$. At this point it is constructive to discuss some examples:

1. *Addition:* Suppose $f(a, b) = a+b$; this is not a very interesting secure protocol, because when given $a + b$ and a, Alice can deduce b. Note that this is more interesting if there are multiple parties.

2. *Scalar Product:* If f is the scalar product of two vectors, then it does not reveal to either party the other party's vector. However, if Alice can perform n such computations (where n is the size of the vector) using linearly independent inputs while Bob uses the same vector, then she can learn his vector. However, there are situations where a secure protocol for scalar product is useful.

3. *Set Intersection:* If f is the intersection between two sets, then in most cases the output does not reveal one party's input to the other party. However, if Alice inputs the universe of all items, then she will learn Bob's set. Hence, a secure protocol for set intersection should be used only when the universe of items is prohibitively large, or if this type of misbehavior can be detected. There are many situations where a secure protocol for set intersection is useful (as will become apparent later in this document).

2.2 Definitions and Notations

In this Section we introduce various terms used in this thesis and in the literature of secure function evaluation.

2.2.1 Models of Behavior for Secure Computations

Two-party vs. Multi-party: Secure computation can involve either two parties, or multiple parties. The work in this thesis is mostly two-party based, but there are some exceptions. In either case we model the adversary as a single entity that controls a subset of the parties.

Semi-Honest Adversaries: A semi-honest (also known as "honest-but-curious" or passive) adversary engages in the specified protocol exactly, but he will try to compute additional information from the transcript of the protocol. In this thesis, our primary focus is this adversary model, because if one cannot solve a problem in this simplified model in a practical matter, then there is no hope that a practical protocol could be developed for a stronger adversary model.

Malicious Adversaries: A malicious adversary can deviate arbitrarily from the protocol to either influence the outcome of the protocol or to gain additional information. In some cases it is regarded as a violation of security if a set of parties can force the protocol to terminate. However, if the adversary does not consist of a strict minority of the parties, then preventing this behavior is impossible (as this is a natural consequence of the general impossibility results proved for Byzantine agreement [6]). Note that this implies that early termination cannot be prevented in two-party computation.

Adaptive Adversaries: An adaptive adversary can corrupt a different set of parties during each round of the protocol (that is the malicious set can dynamically change as the protocol unfolds). In this thesis, we focus entirely on non-adaptive adversaries.

Computationally-Bounded Adversaries: In this thesis, we consider only adversaries that are computationally-bounded. That is, we model adversaries as probabilistic polynomial time algorithms.

2.2.2 Definitions

In this Section, we informally describe various terms that are used in the remainder of this Chapter. To avoid cluttering this exposition with unnecessary formalism, we omit formal definitions for these terms, but the reader is referred to [7,8] for such definitions.

1. *One-way functions*: A one-way function is a function that is easy to evaluate but is difficult to invert. A similar notion is that of a cryptographic hash function.

2. *Pseudorandom functions*: A pseudorandom function is a function whose output is indistinguishable from a randomly generated value.

3. *Trapdoor functions*: Trapdoor functions are similar to one-way functions in that they are easy to evaluate and are hard to invert. However, if one knows certain "trapdoor" information, then it is possible to invert the function.

4. *Negligible:* A function v is negligible if for every positive polynomial p and sufficiently large n, $v(n) < \frac{1}{p(n)}$.

5. *Rounds:* There are some discrepancies in the literature as to what consists of a round in a protocol. It is either a single message between Alice and Bob or is a roundtrip (i.e., it is a message from Alice to Bob and then another message from Bob to Alice). In this thesis, we use message exchange to describe the former and round to describe the latter.

2.3 A Formal Definition

We intially state a formal definition of security for semi-honest adversaries in two-party computation, and then later extend it to malicious adversaries and multi-party computation.

2.3.1 Semi-Honest Adversaries

In this Section, we describe our security model, which is derived from the standard definitions in the literature [2, 4, 5]. Recall that a protocol securely implements a function f if the information that can be learned by engaging in the protocol, could be learned in an ideal implementation of the protocol (i.e., where the functionality is provided by a trusted oracle). We consider semi-honest adversaries (i.e., those that will follow the protocol but will try to compute additional information other than what can be deduced from their input and output alone).

Before we review the formal definition of secure protocols, we review computational indistinguishability [8]. The notion of computational indistinguishability is used to state that two objects are equivalent from a computational point of view: Two objects are equivalent to a computationally bounded distinguisher, if no probabilistic polynomial time (PPT) algorithm can distinguish them. We use a standard notion in computational complexity by looking at infinite sequences of strings. Specifically, we consider infinite sequences of distributions, $\{X_n\}_{n\in\mathcal{N}}$; such a set of values is called a probability ensemble. Two probability ensembles, $\{X_n\}_{n\in\mathcal{N}}$ and $\{Y_n\}_{n\in\mathcal{N}}$ are computationally indistinguishable if for any PPT algorithm D and any positive polynomial p, it holds that:

$$|(Pr(D(X_n, 1^n) = 1)) - (Pr(D(Y_n, 1^n) = 1))| < \frac{1}{p(n)}$$

In what follows, we define an ideal model (with TTP) and a real model (without TTP) for computing a function f; we will establish security by showing that any behavior that is achievable by an adversary in the real model is also achievable by an adversary in the ideal model. We use X_A and X_B to represent Alice and Bob's respective inputs, and we assume that Bob is honest (a symmetrical definition follows when Alice is honest).

In the ideal model, Alice (Bob) is viewed as a Probabilistic Polynomial Time (PPT) algorithm A (B) that can be decomposed into two parts A_I and A_O (B_I and B_O). An ideal-model adversary is admissible if it corrupts at most one of the parties. Alice's view of the protocol is represented by $\mathit{IDEAL}_{A,B}(X_A, X_B) = (A_O(X_A, r_A, f(A_I(X_A, r_A), X_B)), f(A_I(X_A, r_A), X_B))$ where r_A denotes Alice's private coin flips.

Suppose that the real model uses a protocol Π to compute the function f. In the real model, Alice and Bob are modeled by arbitrary PPT algorithms A' and B'. An adversary is admissible if both parties use the algorithm specified by protocol Π (recall we are in the semi-honest model). We denote the adversary's view of protocol

Π by $\mathit{REAL}_{\Pi,A',B'}(X_A, X_B)$, which is the output from the interaction of $A'(X_A)$ and $B'(X_B)$ for protocol Π.

A protocol Π securely evaluates a function f if for any admissible adversary (A', B') in the real model, there exists an admissible ideal-model adversary (A, B) such that $\mathit{IDEAL}_{A,B}(X_A, X_B)$ and $\mathit{REAL}_{\Pi,A',B'}(X_A, X_B)$ are computationally indistinguishable.

2.3.2 Malicious Adversaries/Multiple Parties

We now extend the previous definition to the malicious model. The only required change to the definition is that a real model adversary is said to be admissible if at least one of the parties use the algorithm specified by protocol Π; the other party is modeled by an arbitrary PPT algorithm.

To extend this definition to multiple parties, one needs to define an algorithm to represent the behavior of each party. We represent the adversary by a single PPT algorithm that controls the malicious subset of the parties.

2.4 Literature Review

In this Section, we briefly describe the relevant previous work in Secure Function Evaluation (SFE) and Secure Multiparty Computation (SMC).

2.4.1 Two-party Computations

In [9] Yao introduced the notion of a secure protocol. Specifically, his work focused on comparing two values (called Yao's Millionaire Problem). His technique can be extended to securely compute any function, but his original scheme requires exponential communication.

In [10], Yao introduced a semi-honest constant-round protocol. His scheme allows the parties to securely evaluate any binary circuit with communication and computation proportional to the size of the circuit. Thus, any function in P can be computed with polynomial communication and computation. We describe this scheme in more detail in Section 3.4. Recently, Malkhi et al. [11] implemented Yao's protocol and showed that it is practical for some problems. Cachin et al. [12] introduced a single round version of this protocol.

The above schemes are resilient only against semi-honest adversaries. Goldreich et al. [13] introduced a malicious model protocol for two parties. However, this protocol is not a constant-round protocol. Lindell [14] introduced the first constant-round version of general two-party secure function evaluation under the malicious model. In [15], Katz and Ostrovsky introduced a constant round protocol for secure function evaluation in the malicious model. They show that 5 messages exchanges is enough for any secure protocol in the malicious model, and they prove that there does not exist a 4 message exchange secure protocol for computing any function in the malicious model.

2.4.2 Multi-party Computations and Stronger Adversary Models

Goldreich et al. [13] introduced protocols for malicious adversaries and multiple parties. They introduced protocols for evaluating any function in the malicious model. However, this protocol is impractical for many problems. There have been several improvements to this work [16–29]. Many protocols for SMC have been developed for stronger adversary models. These models include: Asynchronous networks [30, 31], adaptive adversaries [32, 33], generalizations of the adversary structures [34], and SMC in the presence of quantum inputs [35].

2.4.3 Alternate Models

There have been many alternate models under which SMC has been studied. We now describe those that impact this thesis.

Feige et al. [36] introduced a minimal model for secure computation. In their model, they introduce a third party that is untrusted (i.e., it should not learn the inputs of the parties nor the result). They showed that if the adversary is computationally bounded and corrupts at most one party, then there is single-round secure protocol with polynomial communication and computation for any function in P. Beaver [37] introduced a similar model called the commodity based model. In this model there are many servers (instead of one) that are untrusted (but are trusted not to collude with the participants). Their protocols do not require the servers to interact with other servers, is secure against any coalition of servers, and is secure even if a minority of the servers are corrupted by one of the participants.

Sander et al. [38] introduced the notion of cryptocomputing. In cryptocomputing, Alice has a value x, Bob has a circuit C, and at the end of the protocol, Alice learns $C(x)$. Their protocols requires communication and computation exponential in the depth of the circuit, and thus this protocol requires polynomial communication and computation as long as Bob's circuit is in NC^1 (i.e., log-depth circuits). Furthermore, Alice does not learn anything about C other than what can be deduced from C's size and depth and the result. Beaver [39] extended the results of [38] to every function in NLOGSPACE (Non-deterministic log space).

Naor and Nissim [40] studied a new model for general SMC. Instead of using circuits to compute the function, they propose a protocol based on the communication complexity tree for the function. They introduced protocols for computing a function with communication complexity c securely with communication that is polynomial in c but with computation that is exponential in c.

2.4.4 Composition Theorems

While it is not always true that two secure protocols can be composed to create another secure protocol, Canetti [41] gave conditions where composition is valid (i.e., preserving security). Suppose a protocol Π for evaluating a function g invokes ideal implementations (i.e., using a trusted third party) of functions $f_1, f_2, \ldots, f_n$ and is proven secure. Then a protocol Π' for computing g that replaces the ideal implementations of the f_i functions by secure evaluations, is also secure. This composition theorem holds for many adversary models including semi-honest, malicious, and adaptive adversaries.

2.4.5 Random Oracle Model

Bellare and Rogaway [42] introduced the Random Oracle Model as a technique to prove that various protocols are secure. It assumes that all parties have access to a shared random oracle (i.e., $G : \{0,1\}^\star \rightarrow \{0,1\}^k$ where k is some fixed value). In other words, the parties have a shared server that provides a random function. A protocol is developed using this oracle and is proven secure. The protocol is implemented by replacing the calls to the random oracle with invocations of a one-way function.

Canetti et al. [43] showed that there exist signature and encryption schemes that are secure in the Random Oracle Model, but when the oracles are replaced by any implementation of an oracle the schemes are not secure. While this negative result casts suspicion on the Random Oracle Model, this model is still widely assumed and is believed to be valid for many cases. Clearly, a protocol that does not assume the Random Oracle Model is preferred over one that does assume it.

2.5 A Survey of Specific Secure Protocols

In this Section, we describe several specific secure protocols. These protocols include: oblivious transfer, private information retrieval (PIR), set intersection, finding the kth ranked element, Yao's millionaire problem, and secure auctions.

2.5.1 Oblivious Transfer

The notion of oblivious transfer (OT) was introduced by Rabin [44], and one of its first uses was described in [45]. There are many variations of OT. The original notion of OT was that the sender has a single message, and the receiver learns this message with probability one half. A variation of this scheme was described by Even et al. [46], where the sender has two messages and the receiver obtains exactly one of these messages with the probability of receiving a specific message being exactly one half. Another variation, all-or-nothing disclosure of secrets (or chosen OT), was introduced by Brassard et al. [47]. In chosen OT, Alice has N messages and Bob gets to choose exactly one of the messages, he learns no information about any other messages, and Alice learns no information about which message Bob chose. Combining the results of Brassard et al. [48] and Crépeau [49], leads to the surprising conclusion that all of the previous forms are equivalent. That is if one of them exists, then they all exist.

Impagliazzo and Rudich [50] showed that there does not exist a black-box reduction from OT to one-way functions. They did this by showing that the existence of such a reduction would prove $P \neq NP$, and thus it is unlikely that such a reduction exists. This is interesting for a couple of reasons: i) it shows that OT probably cannot be implemented in a way that does not require some form of expensive cryptography (such as public key), and ii) there are many problems that OT can be reduced to, and this implies a level of difficulty for such problems.

There has been other work on developing efficient OT protocols including: [51–53].

2.5.2 Private Information Retrieval

Private Information Retrieval (PIR) was introduced by Chor in [54]. PIR involves a client and a server (note that the server can be multiple replicated servers). The server has a n bits $x_1, \ldots, x_n$ and the client has an index $i \in [1, n]$. At the end of the protocol the client learns x_i and the server learns no information about i. This problem may appear to be equivalent to chosen OT, but in PIR the client can learn more bits than just x_i, which is not the case in chosen OT (that is, the other x_i's are not protected).

Chor et al. [54] studied PIR in an information theoretic sense (i.e., the server learns nothing about the index of the client). They proved that any single server solution must require at least n bits of communication. However, when there are multiple servers a significant savings can be achieved: Chor et al. [54] showed that: i) when there are two databases $O(n^{1/3})$ communication is possible, ii) when k servers are used, $O(n^{1/k})$ communication is possible, and iii) when $(1/3)\log n$ servers are used, polylogarithmic communication is possible.

Chor and Gilboa [55] studied a less stringent notion of security: Instead of information theoretic security, they studied computational security (i.e., resilience against adversaries that are restricted to probabilistic polynomial time). They showed that with two servers, there is a PIR scheme that has communication complexity $O(n^{\epsilon})$ for any $\epsilon > 0$. Kushilevitz and Ostrovsky [56] showed that if the model of security is computational then replication is not needed (i.e., a single server solution with sub-linear communication is possible). They introduced a single-server PIR scheme that has communication complexity $O(n^{\epsilon})$ for any $\epsilon > 0$. A polylogarithmic communication single server PIR solution was proposed by Cachin et al. [57].

Gertner et al. [58] studied a variation on PIR, called Symmetric PIR (SPIR). In this model, it is required that the client learn only a single bit from the server. Note that this is very similar to chosen OT; the principal difference is that in SPIR there can be more than one server. They introduced a SPIR scheme with k servers that requires $O(\log n \cdot n^{1/2k-1})$ communication and a solution with $\log n$ servers that requires polylogarithmic communication.

2.5.3 Set Intersection

Suppose Alice and Bob each have a private set of n elements, and that they would like to determine the intersection between their sets. A simple circuit for the set intersection problem requires $O(n^2)$ gates and thus a protocol that simulates this circuit requires $O(n^2)$ communication.

Huberman et al. [59] gave a two-party protocol that requires only $O(n)$ communication and $O(n)$ modular exponentiations and $O(1)$ rounds. However, this protocol was only analyzed for semi-honest adversaries with the Random Oracle Model. Protocols for private computation of set intersection cardinality for multiple parties have been given by Vaidya and Clifton [60, 61]. These protocols compute the cardinality of the intersection set size. However, the solutions leak the cardinality of intersection of two party's sets to some other party and the protocols given are not resilient against collusion of even two parties.

The state of the art in this work is Freedman et al. [62]. Their contributions are summarized below:

1. A two-party set intersection protocol for the semi-honest model that requires $O(n)$ communications and $O(n \log \log n)$ modular exponentiations without assuming the Random Oracle Model. Protocols are given that compute: i) the actual intersection, ii) the cardinality of the intersection, and iii) whether the cardinality of the intersection is above a threshold.

2. A two-party protocol for the malicious model that is provably secure in the Random Oracle Model.

3. A protocol that approximates the intersection that can be more efficient than the exact solution. However, this protocol does leak some additional information.

4. The protocol is extended to multiple parties.

5. A protocol is given for fuzzy set intersection, where items are said to be a match if they are “close” (e.g., a pair of items match if 3 of their 5 attributes are a match).

2.5.4 Finding the kth Ranked Element

Suppose Alice and Bob have respective datasets D_A and D_B, which consist of elements from a totally ordered domain. The kth ranked element is the item in $D_A \cup D_B$ that has exactly $k-1$ items in $D_A \cup D_B$ which are smaller than it. Of course, this can be generalized to more than two parties, and we can assume that $|D_A| \leq k$ and $|D_B| \leq k$ (since Alice and Bob can truncate their datasets at the kth item).

Suppose that there are k items in both sets, and that each item is represented by $\log M$ bits. A naive circuit for finding the kth ranked elements would require $O(k)$ comparisons of size $O(\log M)$. Thus, the circuit-based approach for this problem requires $O(k \log M)$ communication. A solution based on communication complexity requires $O(k)$ computation. Aggarwal et al. [63] introduced solutions with sub-linear communication costs. Their solution is based on a well-known $O(\log k \log M)$ solution based on communication complexity [64], and requires $O(\log k \log M)$ communication for two-party computation of the kth ranked element without duplicate items in the semi-honest and malicious models . Aggarwal also introduced a multiple party protocol that supports duplicate items that requires only $O(\log^2 M)$ communication.

2.5.5 Yao’s Millionaire Problem

Yao’s Millionaire Problem [9] involves two “millionaires” who want to determine who has more money without revealing their individual amounts. This involves computing a comparison function. The communication complexity of comparing two n-bit numbers is $\Omega(n)$. Thus, Yao’s circuit simulation [10] is asymptotically optimal in terms of communication. The problems with this solution is that it involves $O(n)$ modular exponentiations and extending it to the malicious model is expensive.

Cachin introduced an improved comparison protocol [65]. His scheme utilized an oblivious third party that did not learn information about either party's inputs and did not learn the result of the computation. This third party was trusted not to collude with either of the participants of the protocol. Cachin's protocol requires each party to perform $O(n)$ public key operations, but is secure in the malicious adversary model when at most one party is corrupt.

Fischlin introduced another protocol for Yao's Millionaire Problem [66]. His protocol is probabilistic, and requires only $6n\lambda$ modular multiplications where the probability of an error in the protocol is bounded by $5n2^{-\lambda}$.

2.5.6 Secure Auctions

There are many issues when developing a secure auction service (such as how to ensure that the winner actually pays, etc.). A first work in this area was [67]. This work assumes that bids could be revealed after the bidding phase of the auction was closed. However, in many cases it is desirable to compute the winning bidder(s) along with the bid value(s) without revealing other information.

One approach for computing these values is for the bidders to engage in a secure protocol. Such a scheme requires a threshold trust assumption (i.e., it is assumed that a majority of bidders are honest). Some examples of this approach are Harkavy et al. [68] and Kikuchi [69]. Other protocols do not require threshold trust. Of course this comes at the cost of robustness (i.e., nodes can force early termination), but this misbehavior is detectable and it is assumed that such misbehavior is punished outside of the protocol (by fines perhaps). Thus this is a reasonable model for some auctions. Some examples of such schemes are: Brandt [70, 71] and Brandt and Sandholm [72]

Another approach is to have an untrusted third party engage in a protocol with the seller, so that neither the seller nor this third party learn any additional information. In many cases, this third party can be chosen so that it has nothing to gain from begin dishonest. A benefit of this model is that the bidders only have to submit their bids, as compared to the previous model where they must engage in a protocol. Example schemes include: Baudron and Stern [73], Lipmaa et al. [74], and Naor et al. [75].

3 BUILDING BLOCKS

In this Chapter we introduce the building blocks used in the rest of this thesis; two of these building blocks are novel contributions of this thesis. The building blocks described in this Section include: encryption, split data, oblivious transfer, scrambled circuit evaluation, set intersection, scalar product, comparing modularly split values, and reducing the base of modularly split values.

3.1 Encryption

In this Section, we describe the types of encryption that are used in this thesis.

3.1.1 Symmetric Key Encryption:

We use Enc to denote the encryption algorithm of a symmetric key encryption scheme (such as AES), and Dec to denote the corresponding decryption algorithm. Let M be a message and k be a key for the symmetric key encryption scheme. We use $Enc(M, k)$ to denote the encryption of M with the key k, and we use $Dec(M, k)$ to represent the decryption of M with the key k. By definition of encryption and decryption: $Dec(Enc(M, k), k) = M$.

3.1.2 Homomorphic Encryption:

Another type of encryption used in this thesis is homomorphic encryption. A cryptographic scheme with encryption function E is said to be homomorphic if the following holds: $E(x) * E(y) = E(x + y)$. Another property of such a scheme is that $E(x)^y = E(xy)$. The arithmetic performed under the encryption is modular, and the modulus is part of the public parameters for this system. Homomorphic schemes are described in [76–78]. Typically, we utilize homomorphic encryption schemes that are semantically secure. A scheme is *semantically secure* if when given the public parameters to a homomorphic scheme E, and either $E(m)$ for a specific message m or $E(m')$ where m' is chosen uniformly from the message space, then for any PPT algorithm P, $|(Pr(P(E(m))) = 1) - Pr(P(E(m')) = 1)|$ is negligible. To restore the semantic security of a specific encrypted value, it is enough to multiply by $E(0)$, which refreshes the randomness of the encryption but does not change the encrypted value.

3.1.3 Identity-Based Encryption:

The concept of Identity-Based Encryption (IBE) was first proposed by Shamir [79] in 1984, however the first usable IBE systems were discovered only recently [80, 81]. An IBE scheme is specified by the following four algorithms:

1. *Setup:* A Private Key Generator (PKG) takes a security parameter k and generates system parameters params and a master secret s. params is public, whereas s is private to PKG.

2. *Extract:* Given any arbitrary $\mathsf{ID} \in \{0,1\}^*$, PKG uses params, s, and ID to compute the corresponding private key d_{ID}.

3. *Encrypt:* It takes params, ID and plaintext M as input and returns ciphertext C.

4. *Decrypt:* It takes params, d_{ID} and ciphertext C as input and returns the corresponding plaintext M.

An IBE scheme enables Bob to encrypt a message using Alice's ID as the public key, and so he avoids obtaining the public key from Alice or a directory. Boneh and Franklin proposed an IBE scheme from the Weil pairing [80]. Their scheme is secure against adaptive chosen ciphertext attacks and is semantically secure (note that this last property is useful because it states that an adversary cannot learn which identity has been used to encrypt a specific message when given the ciphertext).

3.2 Split Data

Sometimes it is desirable to store items in a split manner (i.e., where neither party knows the values, but each party has a share of the value). This allows for a straightforward composition of secure building blocks into secure protocol using composition theorems [41]. In this thesis, we use three types of split data: additive, modular additive, and exclusive-or (XOR). Whenever a value x is split, we denote the respective shares by x' and x'', where $x = (x' + x'', x' + x'' \bmod M, x' \oplus x''$ for additive, modular additive, and XOR split respectively). A problem with additively split data is that it is probabilistically leaky, but it allows for comparison using standard techniques (see [82]). Later in this Chapter, we introduce protocols for comparing modular-additively split values.

3.3 Oblivious Transfer

Recall Oblivious Transfer from Section 2.5.1. We use 1-out-of-n Oblivious Transfer (OT) as a basic building block in our protocols. In this thesis, we use chosen 1-out-of-n OT protocol exclusively. Recall that in this variation there is a sender with n private messages $\langle m_0, m_1, \ldots, m_{n-1}\rangle$, and a chooser with an index $i \in \{0, \ldots, n-1\}$. At the end of the protocol, the chooser obtains m_i without learning anything else, while the sender learns nothing about i.

3.4 Scrambled Circuit Evaluation

We now review Yao's scheme for secure two party circuit simulation in constant rounds [10]. In this protocol, one party is a *generator* of a scrambled circuit and the other party is an *evaluator*. The generator creates a scrambled circuit where each wire of the circuit has two encodings and the evaluator learns the encoding for the value of each of the circuit's wires. This a secure circuit evaluation, because the evaluator learns only a single encoding per wire and does not know what this encoding means.

As it is necessary to understand some of our later protocols, we now describe the details of Yao's protocol for Scrambled Circuit Evaluation.

- Circuit Generation: For each wire in the circuit $w_1, \ldots, w_n$, the generator creates random encodings for the wires (in this case the encodings are keys for a trapdoor function or are a random seeds that can be used by a pseudo-random number generator to generate such keys). We denote the encodings of 0 and 1 for wire w_i by $w_i[0]$ and $w_i[1]$ respectively. In order to evaluate the circuit, gate information must be constructed that allows the evaluator to compute the encoding of a gate's output wire when given the encodings for this gate's input wires. The gate information for a 2-ary gate that computes a function f with input wires w_i and w_j and with output wire w_k consists of four messages (where q is a publicly defined marker, used to recognize when an item has been successfully decrypted):

    1. $Enc(Enc(q||w_k[f(0,0)], w_j[0]), w_i[0])$
    2. $Enc(Enc(q||w_k[f(0,1)], w_j[1]), w_i[0])$
    3. $Enc(Enc(q||w_k[f(1,0)], w_j[0]), w_i[1])$
    4. $Enc(Enc(q||w_k[f(1,1)], w_j[1]), w_i[1])$

    Note that the information for a gate consists of these messages in a randomly permuted order (where this order is known only to the generator). Clearly, the scrambled circuit with fan-in 2, can be represented in size proportional to the size of the circuit. The above construction can easily be modified to handle n-ary gate with m outputs, but where the gates have size $2^n m$. While this should be avoided for large n due to the exponential blowup in gate size, there are situations where this is useful.

- Learning Input Wires: In order to evaluate a circuit the evaluator must know the encodings of the input wires. For input wires corresponding to the generator's inputs, the generator simply sends the evaluator the encoding for each of his inputs. For input wires corresponding to the evaluator's inputs, the two parties engage in a chosen 1-out-of-2 Oblivious Transfer(OT) where the two "messages" are the generator's encodings for 0 and 1, and the evaluator chooses the encoding corresponding to his input for that wire.

- Evaluating the Circuit: To evaluate a gate, the evaluator decrypts each value of the gate's information with the keys that it has for the input wires. Only one of these decrypted messages will contain the marker q (the others will look random), and thus the evaluator will learn exactly one encoding for the output wire (he will know that it is the correct value for that wire, but of course he cannot tell whether it corresponds to a 0 or a 1).

- Learning the Result: If the goal is to have the evaluator simply learn the result, then it is enough for the generator to tell the evaluator both encodings for the output wires.

Note that the above protocol for Scrambled Circuit Evaluation can securely evaluate a 2-ary circuit with m gates and n input wires with $O(n)$ 1-out-of-2 OTs, $O(m)$ communication proportional, $O(m)$ evaluations of a trapdoor function, and $O(1)$ rounds; specifically this scheme requires 2 rounds.

3.5 Scalar Product

This protocol allows Alice and Bob to compute the scalar (i.e., "dot") product of two vectors. It is denoted by $SCALARPRODUCT(\vec{v_1}, \vec{v_2})$ where Alice has one vector and Bob has the other (alternatively both vectors can be modular additively split between Alice and Bob). Secure protocols for scalar product were proposed by Du and Atallah [83] and by Vaidya and Clifton [84]. However, Goethals et al. [85], showed that these schemes have security vulnerabilities. Goethals et al. also introduced a scheme based on semantically-secure homomorphic encryption that requires a single round and where each party performs $O(|v_1|)$ modular exponentiations.

3.6 Set Intersection

Freedman et al. [62] introduced an elegant scheme for computing the intersection of two k element sets with only $O(k)$ communication and $O(k \ln \ln k)$ modular exponentiations. We use a simple variation of this scheme where Alice inputs a set of values S_A and Bob inputs a specific value x along with a message M. At the end of the protocol, Alice learns M if her set contains x and learns a random value otherwise. This primitive can easily be defined from the protocols in [62] and requires a single round of interaction between Alice and Bob, $O(\rho|S_A|)$ communication (where ρ is the size of the security parameter), $O(|S_A|)$ modular exponentiations performed by Alice and Bob, and is secure against a malicious Alice.

3.7 Comparing Modular Values

In some of our protocols the intermediate results are stored in modular additively split fashion between the two parties. There are two difficulties with such an approach: i) comparing these values and ii) many times the modulus is very large (much larger than the split value) which leads to inefficient protocols. We introduce protocols for comparing modular additively split values (Section 3.7.1) and for reducing the modulus of such values (Section 3.7.2).

3.7.1 Comparison

While it is difficult to securely compare modular additively split values efficiently in all cases, when the modulus is more than double the largest possible value being compared it is roughly twice as difficult (in terms of communication and computation) as a standard secure comparison. Figure 3.1 introduces a protocol for comparing two modular additively split values when the above conditions hold; we are not aware of such a protocol elsewhere.

Input: Alice has two values x' and y' and Bob has two values x'' and y''; all of these values are in the range $[0, M)$. Furthermore the sums $(x' + x'') \bmod M$ (i.e., x) and $(y' + y'') \bmod M$ (i.e., y) are in the range $[0, m)$. It is also known that $M \geq 2m$.
Output: Alice and Bob would like to compute in XOR-split fashion whether or not $(x'+x'') \bmod M \leq (y'+y'') \bmod M$. Note that the protocol below can easily be modified to support other types of comparison.
Notes: All arithmetic in the protocols is modulo M.

1. Alice computes $a \leftarrow y' - x' - m + 1$, $b \leftarrow y' - x'$, and $c \leftarrow \mathit{false}$. If $a \geq b$ (i.e., there is wrap-around), then Alice sets her values to $a \leftarrow y' - x' + 1$, $b \leftarrow y' - x' + m - 1$, and $c \leftarrow \mathit{true}$. Bob computes $d \leftarrow (x'' - y'')$.
2. Alice and Bob engage in a protocol using Scrambled Circuit Evaluation to evaluate $(d \geq a) \wedge (d \leq b)$ in an XOR-split fashion, thereby obtaining r_A and r_B. Alice and Bob's respective outputs are $r_A \oplus c$ and r_B.

Figure 3.1. Protocol for Comparing Modularly-Split Values

Before we prove the correctness of the above protocol, we give two examples which will clarify it. For both of the examples, suppose that $m = 4$ and the $M = 8$ (satisfying $M \geq 2m$). In what follows when we define ranges $[a, b]$, we are referring to a range modulo M with the possibility of wrap-around and all arithmetic is done modulo M (e.g., the range $[6, 1] \equiv \{6, 7, 0, 1\}$).

Example 1 Suppose $x' = 3$ and $y' = 2$. Alice knows that Bob's values satisfy: $x'' \in \{5, 6, 7, 0\}$ and $y'' \in \{6, 7, 0, 1\}$. The reader can easily verify that, after step 1 of the protocol, (a, b, c) is $(4, 7, \mathit{false})$ and $(x \leq y) \equiv (x'' - y'' \in [4, 7])$.

Example 2 Suppose $x' = 3$ and $y' = 4$. Alice knows that Bob's values satisfy: $x'' \in \{5, 6, 7, 0\}$ and $y'' \in \{4, 5, 6, 7\}$. The reader can easily verify that, after step 1 of the protocol, (a, b, c) is $(2, 4, \mathit{true})$ and $(x \leq y) \equiv (x'' - y'' \in [6, 1])$, which is equivalent to $(x \leq y) \equiv (x'' - y'' \notin [2, 4])$.

Proof of Correctness: Alice knows that: i) $x'' \in [-x', -x' + m)$ and ii) $y'' \in [-y', -y' + m)$. Suppose the values of x and y are respectively i and j, then $x'' =$

$-x'+i$ and $y'' = -y'+j$. Clearly, $x''-y'' = y'-x'+(i-j)$. Thus, $(x \leq y) \equiv x''-y'' \in [y'-x'-m+1, y'-x']$ and similarly $(x \leq y) \equiv x''-y'' \notin [y'-x'+1, y'-x'+m-1]$. Since $M \geq 2m$ these ranges must be disjoint and thus at most one of the ranges has wraparound. For a range $[a,b]$ that does not wraparound it is easy to determine if a value z falls in the range by checking if $z \geq a$ and $z \leq b$. The above clearly mimics what the protocol does. □

Proof of Security: Clearly, the first step of the protocol does not reveal any information (as the computations are done locally and there is no communication). If the implementation of Step 2 was provided by a trusted oracle, then this protocol would be secure. Since Scrambled Circuit Evaluation will securely evaluate the second step, this protocol is secure by the composition theorem in [41]. □

3.7.2 Base Reduction

Another problem with using modular additively split values is that many times such values are modulo the base of a homomorphic encryption scheme (i.e., as in a scalar product protocol with homomorphic encryption). And if the values are compared with the technique in the previous Section, then the communication is proportional to the number of bits in the homomorphic base, which is typically much larger than the number of bits required to represent the value. Figure 3.2 presents a protocol for reducing the base of a modularly split value (this protocol is not used explicitly in our protocols as it is not required for correctness, but it will decrease the communication for many of our protocols at the cost of an additional round of communication). We are not aware of this protocol elsewhere.

Before we prove the correctness of the above protocol, we give two examples which will clarify it. For both of the examples, we suppose that $m = 4$ and the $M = 9$ (satisfying $M \geq 2m$).

Example 3 Suppose $x' = 3$ and $x'' = 0$. Alice computes $y' = 3 - r$ for some random r, and her list is set to $(v_0, v_1) = (r, r-1)$. Since $0 \leq 3$, Bob will choose

Input: Alice has a value x' and Bob has a value x''; both of these values are in the range $[0, M)$, but $(x' + x'') \bmod M$ (i.e., x) are in the range $[0, m)$. It is also known that $M \geq 2m$.
Output: Alice and Bob have respective values y' and y'' in the range $[0, m)$, such that $(x' + x'') \bmod M = (y' + y'') \bmod m$.
Steps:

1. Alice generates a random value r chosen uniformly from the range $[0, m)$. Alice computes $a \leftarrow (M \bmod m)$. Alice sets her output to be $y' \leftarrow ((x' \bmod m) - r) \bmod m$. Alice creates a list of two values v_0, v_1: If $x' \geq m$, then she sets both values to $(r - a) \bmod m$, otherwise she sets the values to r and $(r - a) \bmod m$.
2. Bob computes a Boolean value $i \leftarrow (x'' > m)$, where i is either 0 or 1. Alice and Bob engage in a chosen 1-out-of-2 OT where Bob learns v_i. Bob sets his output to be $y'' \leftarrow (x'' + v_i) \bmod m$.

Figure 3.2. Protocol for Reducing the Base of Modularly-Split Values

the first item and thus will obtain r. Alice and Bob's respective outputs are thus 3-r and r, which are correct as their sum is 3.

Example 4 Suppose $x' = 5$ and $x'' = 5$. Alice computes $y' = 1 - r$ for some random r, and her list is set to $(v_0, v_1) = (r - 1, r - 1)$. Since $5 > 3$, Bob will choose the second item and thus will obtain $r - 1$. Alice and Bob's respective outputs are thus 1-r and r, which are correct as they sum to 1.

Proof of Correctness: There are two cases: i) $(x' + x'') \in [0, m)$ (no wraparound) and $(x' + x'') \in [M, M + m)$ (wraparound). In case i, both x' and x'' are smaller than m, and so $y' = (x' - r) \bmod m$ and $y'' = (x'' + r) \bmod m$. Thus, $(y' + y'') \bmod m = (x' + x'') \bmod m$, which is correct. In case ii, suppose $(x' + x'') = M + \beta$. The respective outputs will be, $y' = (x' - r) \bmod m$ and $y'' = (x'' + r - a) \bmod m$. Now, $(x' + x'') \bmod m = (M + \beta + r - r - a) \bmod m$, but since $a = M \bmod m$, this is equal to β. □

Proof of Security: The main concern is that Alice or Bob could learn something about the value of $(x' + x'') \bmod m$ by engaging in the protocol. The only interaction in this protocol is the OT protocol. By definition this reveals nothing to Alice. Bob either obtains r or $r - a \bmod m$, but since r is uniformly distributed in $[0, m)$, this is computationally indistinguishable from a random value. $\square$

4 SECURE VERIFIABLE OBLIVIOUS FUNCTION EVALUATION

In this chapter, we introduce a variation on standard secure function evaluation. What separates this work from traditional SFE is: i) the function that is to be computed is private to one party, ii) the result of the function is typically revealed to only one party (and this does not need to be the party with the function), and iii) one party's inputs must be verified by a third party that is not necessarily online during the protocols. We call this type of computation Secure Verifiable Oblivious Function Evaluation. In this chapter, we present general techniques for achieving the above-mentioned problem, and then apply these techniques to two domains: trust negotiation and credit checking. Although we choose to present the techniques for the trust negotiation and credit checking application domains, they have much broader applicability and in fact work for any situation where there is a repository of public and private information about individuals, that is subsequently used for making decisions that impact the individuals.

The organization of this chapter is as follows: In Section 4.1, we describe trust negotiation domain, and in Section 4.2 we introduce the credit checking domain. In Section 4.3 we describe basic techniques used in both domains. We introduce protocols for trust negotiation and credit checking in Sections 4.4 and 4.5 respectively. Finally, we summarize this chapter in Section 4.6.

4.1 Trust Negotiation

Whereas in the past access decisions were based on the identity of the entity requesting a resource, in open systems such as the Internet, this approach is ineffective when the resource owner and the requester belong to different security domains controlled by different authorities that are unknown to each other. One alternative is to use *digital credentials* for satisfying access policies. Digital credentials, the digital equivalent of paper credentials, are digitally signed assertions about the credential owner by a credential issuer. Each digital credential contain an attribute (or set of attributes) about the owner. The decision to access a resource is based on the attributes in the requester's credentials, such as age, citizenship, employment, group membership, or credit status.

A typical scenario for accessing a resource using digital credentials is for the requester Alice to send her request to Bob, who responds with the policy that governs access to that resource. If Alice's credentials satisfy Bob's policy, she sends the appropriate credentials to Bob. After Bob receives the credentials and verifies them, he grants Alice access to the resource. Observe that, in this scenario, Alice learns Bob's policy and Bob learns Alice's credentials. Such a strategy is straightforward and efficient, however it is unacceptable if the credentials or the access control policies are considered to be sensitive information. In the following, we give a simple example where both the credentials and the policy are sensitive.

Example 5 Consider an online business that grants access to media records by sending access keys to its client's special media-reader software – keys that the reader uses to "unlock" encrypted media records that are freely downloaded in encrypted form (or are given away, in encrypted form, on CDs that are widely distributed for free). Certain records are treated differently from the rest: The online business grants access to these records only if the requester has a disability, or is a senior citizen, or is terminally ill, and has an income of under \$30K a year. This requirement involves four attributes (denote them by $attr_1$, $attr_2$, $attr_3$, $attr_4$) and the policy is $(attr_1 \vee attr_2 \vee attr_3) \wedge attr_4$. In order to gain access to the sensitive records in Bob's database, Alice needs to prove to Bob that she satisfies the policy. However, neither Alice nor Bob is willing to disclose her/his private information. Alice does not want to reveal her credentials, as her credentials contain sensitive information about her (e.g., health, age, income, etc). Bob does not want to reveal the policy, even to those who satisfy the policy, so as to make it harder for an adversary to know which credentials he should forge or otherwise illicitly obtain.

In other examples, the motivation for hiding the policy is not security from an evil adversary, but simply the desire to prevent legitimate users from "gaming" the system – e.g., changing their behavior based on their knowledge of the policy (which can render economically-motivated policy less effective). This is particularly important for policies that are not incentive-compatible in economic terms (they suffer from perverse incentives in that they reward the wrong kinds of behavior, such as free-loading). In yet other examples, the policy is simply a commercial secret – e.g., Bob has pioneered a novel way of doing business, and knowledge of the policy would compromise Bob's strategy and invite unwelcome imitators.

Finally, it is important to point out that a process that protects Alice's credentials from Bob is ultimately not only to Alice's advantage but also to Bob's: Bob no longer needs to worry about rogue insiders in his organization illicitly leaking (or selling) Alice's private information, and may even lower his liability insurance rates as a result of this. Privacy-preservation is a win-win proposition, one that is appealing even if Alice and Bob are honest and trustworthy entities.

In this Chapter, we give efficient protocols that solve the hidden policies with hidden credentials problem. Our protocols are built on the hidden credentials system [86]. The detailed protocols are presented in Section 4.4. In summary, our protocols:

- protect the sensitive credentials – Bob learns nothing about Alice's credentials, not even whether Alice's credentials satisfy the policy.
- protect the policies – Alice learns neither Bob's policy structure nor which credentials gave her access (she only learns whether she satisfies the policy).

4.1.1 Related Work

Our work is related to the area of automated trust negotiation [87–97]. The goal of trust negotiation is to enable strangers to access sensitive data in open environments. In trust negotiation, two parties establish trust through iterative disclosure of credentials and requests for credentials. Most of the research in this area focuses on protecting credentials and assumes policies can be freely disclosed. Some of these [87,88,96] consider access policies as sensitive information. Bonatti and Samarati [87] proposed a framework for regulating service access and information release on the web. Their scheme protects the revelation of policies by dividing policies into two parts: service prerequisite rules, and service requisite rules. A requisite rule is disclosed only after prerequisite rules are satisfied. Seamons et al. proposed the concept of policy graphs in [88]. Instead of using a single policy, their scheme uses policy graphs to represent complex policies and protects polices by gradual disclosure of the graph nodes. Furthermore, Yu and Winslett proposed a unified scheme (UniPro) for resource protection [96]. The basic idea of the UniPro scheme is to model policies as protected resources and protect them in the same way as other resources. Our work offers better protection of policies: in their schemes, Alice learns (part of) the policy if her credentials satisfy the policy, whereas in our protocols, Alice does not learn the policy even if her access request is granted.

Li et al. introduced the notion of Oblivious Signature-Based Envelope (OSBE) [98] to protect sensitive credentials. They assume the content of a credential is non-sensitive (as anyone can come up with it), and only the signature of the credential needs to be protected. In OSBE, Alice sends Bob the content of her credential or a credential that she does not have, and Bob runs an OSBE protocol with Alice, sending an encrypted message to Alice such that Alice can decrypt it if and only if she has the signature on the content. The difference between their work and ours is that the policies in OSBE have to be revealed, whereas the policies in our protocols are protected.

Recently Holt et al. proposed hidden credentials [86], a system that has several remarkable properties besides protecting sensitive credentials and policies. First, their system reduces the network overhead, as it needs fewer rounds of interaction compared to traditional trust negotiation. Second, their system also solves the "going first" problem in PKI-authentication systems, where one of the two parties must be the first to reveal a certificate to a potentially malicious stranger. Our protocols directly build on their work. However, we believe that the protection of policies in their system is not sufficient, for three reasons:

1. The policy structures are revealed in their system. For instance, if Bob's policy is $(attr_1 \wedge attr_2) \vee attr_3$ where $attr_1$, $attr_2$, and $attr_3$ are three attributes respectively, Alice learns the structure of the policy is of the form $(x \wedge y) \vee z$ even if her credentials do not contain any one of the attributes $attr_1$, $attr_2$, and $attr_3$.

2. If an access request to a resource is granted, Alice learns which attributes gave her access. For instance, if Bob's policy is $(attr_1 \wedge attr_2) \vee attr_3$, and Alice's credentials contain $attr_1$ and $attr_2$, and Alice gets the resource, she knows that $attr_1 \wedge attr_2$ is part of the access policy.

3. Even if Alice cannot access the resource, she might learn some partial information about the policy. For instance, if Bob's policy is $(attr_1 \wedge attr_2) \vee (attr_3 \wedge attr_4)$, and Alice's credentials contain $attr_1$ and $attr_3$. Alice learns that $attr_1$ and $attr_3$ are part of Bob's policy.

Bradshaw et al. [99] extended the hidden credentials system to support complex access policies expressed as monotonic Boolean functions. They applied a secret splitting system to conceal the structure of such policies. The extended hidden credentials system protects the structure of Bob's polices, however, it is still unable to solve the problems described in Item 2 and Item 3. However, in our protocols if Alice's credentials match an attribute in Bob's policy, she will not learn that the attribute is part of the policy. In summary, the hidden credentials systems [86] and the trust negotiation systems ([91, 93, 96, 97], to list a few) do not achieve the privacy required by our work. Of course, we do so at a cost in protocol complexity, therefore the present work should be viewed as providing another point on the privacy-performance curve, rather than as an unqualified improvement over the previous work.

4.1.2 Problem Definition

Before we formally define the hidden policies with hidden credentials problem, we first discuss what is an attribute and what is a policy in our problem.

An attribute is a statement about a credential holder. An attribute, for example, could be gender (male or female), job type (student, faculty, FBI agent, etc.), state of residence (California, Indiana, Ohio, etc.), status (secret clearance, disabled, homeless), age (between $[0-17]$, between $[18-20]$, between $[21-59]$, or age ≥ 60), or annual income (between $[0-15K]$, between $[15K-30K]$, between $[30K-60K]$, or income $> 60K$). Let SS be the set of all possible attributes in the hidden credentials system. The CA publishes SS to every user.

Recall each hidden credential binds a username and an attribute. If a user has m attributes, she can get m hidden credentials from the CA, one for each attribute. For instance, if Alice is a professor, she can have a professor credential $cred_1$ where $cred_1.attr =$ professor; if her age is 30, she has a age credential $cred_2$ where $cred_2.attr = age \in [21..59]$ [1].

The goal of our work is to provide privacy-protection for *both* access control policies and credentials. The framework for our *hidden policies with hidden credentials* problem is informally described as follows: Alice has m credentials issued by the CA,

[1] For simplicity, if an attribute takes multiple values such as age and income, we bucket these values into several categories such that it is easier to construct a policy. Consider the policy $age \geq 21$, if we allow age-attribute be any integer value, then the policy would be $age = 21 \vee age = 22 \vee \ldots$, however, if we bucket age-attribute into four groups: $[0-18], [18-20], [21-60], [60+]$, the new policy is simply $[21..59] \vee [60+]$.

denoted as $cred_1, \dots, cred_m$. Each credential contains only one attribute. Bob has a resource M and a policy P for controlling access to M. The policy P is a Boolean function $p(x_1, \dots, x_n) : \{0,1\}^n \rightarrow \{0,1\}$ over n attributes $attr_1, \dots, attr_n$, where $x_i = 1$ if and only if Alice has attribute $attr_i$ in one of her credentials. Function p outputs 0 when Alice does not get access, and outputs 1 when she does. When Alice wants to access M from Bob, she engages in a protocol with Bob. Alice provides the protocol with a subset of her credentials (she may choose to omit certain credentials), whereas Bob provides M and the policy P. If the attributes in the credentials that Alice inputs into the protocol satisfy P, she gets the resource M, otherwise she gets nothing. We want Alice to learn as little as possible about Bob's policy, and Bob to learn as little as possible about Alice's credentials. In the best scenario, Alice only learns whether she satisfies the policy or not, and nothing else (other than what she can deduce from the fact that she gained access), and Bob learns nothing about Alice's credentials or whether she gained access.

More formally, let $\mathcal{P}$ be a family of policy functions. Each $P \in \mathcal{P}$ is a policy function: given any set of hidden credentials C, $P(C) \rightarrow \{0,1\}$. More specifically, a policy function P is a defined by a Boolean function p that is relevant to n attributes $\{attr_1, attr_2, \dots, attr_n\} \subset SS$. Now, $P(C) = p(x_1, x_2, \dots, x_n) : \{0,1\}^n \rightarrow \{0,1\}$, where $x_i = 1$ if $\exists\ cred_j \in C$ such that $attr_i = cred_j.attr$, $x_i = 0$ otherwise.

Let C_A be Alice's hidden credential set. To check whether Alice satisfies a policy P, it is equivalent to verify whether $P(C_A) = 1$. For example, if Bob's policy is $female \wedge student$, i.e., $p(x_1, x_2) = x_1 \wedge x_2$; Suppose Alice has a female credential ($cred_1.attr = female$), a professor credential ($cred_2.attr = professor$), and an Indiana resident credential ($cred_3.attr = Indiana$). Alice does not satisfy Bob's policy, as $P(cred_1, cred_2, cred_3) = p(x_1, x_2) = 1 \wedge 0 = 0$. Note that $x_1 = 1$ because Alice has a female credential, $x_2 = 0$ because Alice does not have a student credential.

We now present the hidden polices with hidden credentials problem. Let M be a private message. Alice and Bob want to compute a function F. Alice has private input C_A, a subset of her credentials[2]. Bob has a private message M and a private access control policy P over M. The function F is defined as follows:

$$\begin{aligned} F_{Bob}(M, P) &= \perp \\ F_{Alice}(C_A) &= \begin{cases} M & \text{if } P(C_A) = 1; \\ \perp & \text{otherwise.} \end{cases} \end{aligned}$$

where F_{Alice} represents Alice's output, F_{Bob} represents Bob's output, and $\perp$ is a special symbol. In other words, our goal is that Bob learns nothing and Alice learns $F_{Alice}(C_A)$ without learning anything else (Alice can infer the result of $P(C_A)$ from her output).

It is possible in our framework to set the policy to be arbitrary, however, it is common in the literature to assume that the policy does not require the absence of a credential because of the practical difficulty of verifying an absence, e.g., if policy is $\neg attr$ then it is possible for Alice who has *attr* to not input the corresponding credential if she suspects it can cause her to be denied access. Furthermore, if Alice knows that her credentials will not be revealed by the protocol and the policy is monotonic then she has an incentive to input all her credentials. It is therefore a practical consideration, rather than an inherent limitation of our scheme, that causes this assumption that the policy does not require the absence of a credential. Sometimes the absence of a credential ("under 21 years of age") can be replaced by a requirement for the presence of the opposite ("over 21 years of age"), but this is not always possible (e.g., consider requiring the absence of a credential for millionaire).

Another issue that needs to be discussed is probing attacks by either party. Alice can engage in the protocol with Bob multiple times using different credential sets (all subsets of her credentials) to gain information about Bob's policy. This type of attack is outside of our model for the problem, however our protocols prevent Alice from probing a policy offline (i.e., requesting a resource once and then trying several subsets of her credentials). Other means for preventing online probing attacks must

[2]Alice may choose a subset of her credentials set instead of inputting all her credentials.

be taken (e.g., make sure Alice can request a resource from Bob no more than three times a week). As Bob does not know whether Alice gained access or not, he cannot carry out probing attacks, although he could if he offered Alice a service rather than a resource: In such a case he could probe Alice's credentials; for example, if Bob sets his policy to be *attr* (secret agent), he can record those who succeed in getting his service and learn who is a secret agent (in such situations Alice would be more conservative when using her sensitive credentials).

4.2 Credit Checking Domain

Typically, when a borrower (Bob) wishes to establish a tradeline (e.g., a mortgage, an automobile loan, or a credit card) with a lender (Linda), Bob is subjected to a credit check by Linda. The reason that Linda does a credit check is so that she will have confidence that Bob is trustworthy enough to pay back the loan. In order for Linda to do a credit check on Bob, she contacts a Credit Report Agency (CRA) and obtains a credit report about Bob. It is the purpose of the CRAs to track financial information about an individual, and currently there are three main CRAs: Equifax [100], Experian [101], and TransUnion [102]. When Linda obtains Bob's credit report, she determines if he qualifies for the loan by determining if his credit report satisfies certain criteria that she defines. There is a substantial amount of information in a credit report including: how many tradelines a person has, how much debt a person owes (and to whom), the number of late payments that a person has made in the past, and many other facts about the borrower's financial history [100–102].

The problem with revealing a borrower's credit report to the lender is that it unnecessarily reveals too much information, often too shady entities who may perform mischief. While many lenders are typically credible organizations, this is not always the case, and even for credible lenders some employees of the lender may be corrupt. Another problem besides the obvious leakage of private information is that this can be an aid for identity thieves as it could serve as a filtering process for them to see who has good credit, and more dangerously, to see who checks their own credit (as this information is contained in the credit report and someone who periodically looks at his own credit report is not as likely to be victimized as someone who never looks at it). Furthermore, lenders do not need to know all of the information in the credit report, but only need to know if certain conditions are satisfied. One naive way to provide privacy would be for Linda to reveal her criteria to the CRA who would then report back to Linda if Bob satisfies her criteria. This is not acceptable for a couple of reasons: i) the CRA becomes a bottleneck of the loan-processing system and ii) Linda's criteria for loan qualification are often confidential.

We propose the usage of privacy-preserving protocols to solve the apparent contradictory goals of being able to determine if a borrower satisfies certain criteria while not revealing private information about the borrower's credit report. After engaging in the protocol, all that Bob should learn is whether or not he qualifies for a loan. Furthermore, all that Linda should learn is whether or not Bob qualifies for the loan. However, there is the additional requirement that the borrower should not be able to lie about his credit report, and thus the information needs to be verified by a CRA. It is also desirable that the information flow of the proposed protocol should mimic the way loan-processing takes place today. All that the CRA should learn is that Bob applied for some line of credit with Linda; of course if Bob qualifies and then accepts the loan from Linda, then she would report the information about the loan to the CRA. With such a set of protocols it would be possible for borrowers to have the option of privacy-preserving credit checking.

In this Chapter, we introduce privacy-preserving protocols for achieving the above-defined problem. The protocols are *efficient* in that they require communication and computation proportional to the size of the credit report and the policy of the lender, and while the computational overhead for the CRA is larger than in the current non-private setting, much of this work can be pre-computed off-line. Furthermore, our protocols are *simple* in that they mimic the way loan-processing takes place today with the only difference being that a borrower must register with the CRA(s) if he wishes to use such protocols. Finally, the protocols are *private* in that they satisfy the privacy requirements outlined above and are *correct* in that the lender has certainty that the borrower's information is accurate.

4.2.1 Problem Definition

The credit report essentially consists of a set of of Boolean values (e.g., "has the borrower ever filed for bankruptcy in the past?") and a set of integers (e.g., the amount of debt owed). A credit report contains other types of information (such as to whom the current debt is owed), but this information is not necessary to make a decision on loan qualification (the type of tradeline might be useful, but this can be encoded as an integer). The credit report can be represented as a set of Boolean *attributes*, where the value is true if a person satisfies that attribute. For Boolean information, the encoding is trivial; for example "has person X ever been bankrupt?" would be an attribute. To encode integer values, an attribute would be defined for each bit in the binary representation of the value; an example in this case would be "is the 5th bit of person X's debt true?". The lender has many *criteria* that it considers for determining loan qualification; the criteria can be computed from one or more attributes. Example criteria include: "does person X have any liens on their home?", "is person X's debt below some threshold?", and "is the number of tradelines that person X has had in the past above some threshold?". If an attribute or a criterion is true for a person, we say that the person *satisfies* the attribute or criterion. We denote the attributes for a credit report by $a_1, \ldots, a_m$ and the criteria by $c_1, \ldots, c_n$, where each criterion is a function of a subset of the attributes. We define a boolean function *sat* where $sat(a_i)$ and $sat(c_i)$ represent whether or not the person satisfies attributes a_i and criterion c_i respectively (i.e., $sat(c_i) = 1$ if c_i is satisfied and is 0 otherwise). Finally, the lender has a *policy* for determining if a borrower qualifies for a specific loan. This policy is some function of $sat(c_1), \ldots, sat(c_n)$.

We make several assumptions in our initial protocol for this problem; many of these assumptions are relaxed in Sections 4.5.3-4.5.7. The assumptions include:

1. *Bounded Credit Report Size:* We assume that each integer value in the credit report can be bounded by some value, and that the total number of entries in the credit report can also be bounded for all people. To protect the size of the credit report the CRA must insert dummy entries into each credit report to make all credit reports not distinguishable by size.

2. *Accurate CRA Assumption:* It is assumed that the CRA is trusted by the lenders to provide accurate information, however the CRA is not a "trusted-third party" in that it should not learn information about a lender's policy. We discuss techniques for handling a malicious CRA that colludes with borrowers to probe a lender's policy in Section 4.5.7.

3. *Single CRA Assumption:* Initially, we assume that there is a single CRA which has all information required by the lender to make a decision about a borrower, however this is not realistic. There may be multiple CRAs with the same information and it is possible that there could be a discrepancy between the CRAs' information. We discuss extensions of our protocols to multiple CRAs in Section 4.5.7.

4. *Criteria Assumption:* We assume that the criteria come in one of two forms: i) a single attribute criteria or ii) a comparison against a threshold criteria. This captures most (if not all) common things done to a credit report by the lender. It is not difficult to extend our protocols to other type of criteria, and we discuss some mechanisms for this in Section 4.5.5.

5. *Known Criteria Assumption:* We assume that the lender does not mind revealing the general form of his or her criteria. For example, the lender is not worried about revealing that it has a criterion that compares a person's debt against a threshold, but the lender does not want to reveal the threshold. In

many cases a global set of criteria may be used by multiple lenders without revealing specific information. This is generalized in Section 4.5.5.

6. *Policy Assumption:* Initially, we assume that the policy is of the form: a borrower qualifies for a loan if he satisfies at least t criteria (where t is a private threshold defined by the lender). While this is not a realistic form for a policy in many cases, it is a preliminary step. We explore generalized policies in Sections 4.5.3, 4.5.4, and 4.5.5.

7. *Passive Behavior Assumption:* Initially, we assume that the parties are passive (i.e., honest-but-curious) in that they will engage in the steps of the protocols, but will try to learn additional information. We generalize our protocols to the malicious adversary model (for borrowers and lenders) in Section 4.5.6. We discuss techniques for handling a malicious CRA that colludes with borrowers to probe a lender's policy in Section 4.5.7.

4.2.2 Privacy and Correctness Requirements

We now define the privacy and correctness requirements of our protocols.

Borrower: All that the borrower should learn is whether or not he qualified for the loan, and nothing else (except of course what he can deduce from this outcome and from his knowledge of his own credit report, which is unavoidable). Furthermore, the borrower should not be able to lie about his credit report (i.e., his values must be the values stored at the CRA).

Lender: The lender should learn whether or not the borrower qualified for the loan, and nothing else (except of course what he can deduce from this outcome and from his knowledge of his own policy, which is unavoidable).

CRA: The CRA should learn only that a specific borrower is applying for a loan with a specific lender. He should learn neither the lender's policy nor whether the borrower qualified for the loan. Of course, in practice, if the loan is approved and does happen, the CRA will be informed by the lender of this fact so he can update the borrower's record – but if the loan is approved yet does not happen for some reason (e.g., the borrower and lender could not agree on an interest rate) then the CRA never learns whether the loan was approved or not.

4.3 Secure Oblivious Function Evaluation

In the construction of a scrambled circuit evaluation, the gates are constructed for some publicly defined function f; to achieve security in the malicious model the generator of the circuit must prove (in zero knowledge) that the circuit is well-formed to the evaluator. However, there are cases where not having the function be publicly defined can be useful. In this case, the standard construction for Scrambled Circuit Evaluation can easily be modified to use *oblivious gates* where the evaluator does not know the function that each gate computes. By doing this, it is possible for the generator to securely evaluate a circuit while only learning its topology (and thus the generator can hide which function is being computed). We now describe three techniques for using this: i) build a topology that can handle many useful functions, ii) use a universal circuit, and iii) use a single n-ary gate for arbitrary functionality. We now explore each of these options in more detail:

4.3.1 Specific Circuits:

We now show that there are topologies of circuits that compute a wide range of useful functions, and thus would allow the generator to evaluate one of many functions. Here are some examples:

1. It is easy to construct an oblivious comparison circuit (i.e., one that can compute $=$, $\neq$, $>$, $<$, $\geq$, and $\leq$ without revealing which comparison is done) with size proportional to the number of bits in the values.

2. A binary tree of oblivious gates (with inputs $a_1, \ldots, a_n$) can be used to compute many useful functions (without revealing which function is being computed) including:

   (a) $\bigwedge_{i=1}^{n} a_i$, $\bigvee_{i=1}^{n} a_i$, $\bigoplus_{i=1}^{n} a_i$, etc.

   (b) For any subset of the values S, $\bigwedge_{i \in S} a_i$,$\bigvee_{i \in S} a_i$, $\bigoplus_{i \in S} a_i$, etc.

   (c) Other functions like: for a subset S_1 of the first half of the values and another subset S_2 of the second half of the values, the function $\bigvee_{i \in S_1} a_i \wedge \bigvee_{i \in S_2} a_i$.

3. By using oblivious binary trees on the results of other binary trees of oblivious gates, a wide variety of policies can be computed including any monotonic circuit and many other useful structures.

4.3.2 Universal Circuits:

Another option is to use a universal circuit, as described by Valiant [103]. Recall that a universal circuit for a specific size s and depth d is a circuit that can be used to evaluate any circuit with these dimensions. Furthermore, there is such a universal circuit of size $O(ds \log s)$ and depth $O(d \log s)$. Thus when such a universal circuit is used, it could represent any circuit of a certain size and depth. Clearly, this could be a wider range of functions than the previous approach, but this comes at a cost to performance.

4.3.3 A Single n-ary Gate:

A final option is to use a single n-ary gate. This scheme allows the function to be arbitrary, but it requires exponential communication. However, any protocol that provides arbitrary functionality must require exponential communication [104].

We have outlined three options for oblivious function evaluation. Furthermore, expressiveness comes as a cost in terms of computation and communication. In the remainder of this Chapter, we will describe how these techniques can be used for trust negotiation and the credit checking domains.

4.4 Protocols for Trust Negotiation

In this Section, we give our protocols for trust negotiation. In Section 4.4.1, we give a review of the hidden credential system that is a basis for our system. In Section 4.4.2 we outline the two phases of our protocols, and then describe protocols for each of these phases in Sections 4.4.3 and 4.4.4. Finally, we give proofs of security in Section 4.4.5.

4.4.1 Review of Hidden Credentials System

In the hidden credentials system proposed by Holt et al. [86], there is a trusted Credential Authorities (CA) who issues credentials for users in the system. Each user in the system is assigned a unique *nym*, where *nym* could be either a real name or a pseudonym. A hidden credential is an encrypted assertion about an attribute of a credential holder by the CA (this behaves similarly to a signature of the credential). Roughly speaking, given an IBE scheme, a hidden credential *cred* for username *nym* and attribute *attr* is the private key corresponding to the identity $nym||attr$. More specifically, the hidden credentials system has the following four programs:

1. $CA\_Create()$: The CA runs the setup program of the IBE system and generates system parameters params and a master secret s, and publishes params. The CA also publishes a list of possible attribute names.

2. $CA\_Issue(nym, attr)$: The CA issues a credential for a user with username *nym* and an attribute *attr* by running the extract program of the IBE system with $\mathsf{ID} = nym||attr$, and outputs the private key d_{ID} as the credential. Given a hidden credential *cred*, we use *cred.nym* to denote the corresponding username, and *cred.attr* to denote the corresponding attribute in the credential.

3. $E_{IBE}(M, nym||attr)$: This program corresponds to the encrypt algorithm of the IBE system with system parameters params, ID $= nym||attr$, and plaintext M. The output of this program is ciphertext C.

4. $D_{IBE}(C, cred)$: This function corresponds to the decrypt program of the IBE system with system parameters params, credential *cred*, and ciphertext C. The output of this function is plaintext M. These programs must satisfy the standard consistency constraint, namely for any credential *cred* and any message M, $D_{IBE}(I(M, cred.nym||cred.attr), cred) = M$. In this paper we assume that, when a user computes $D_{IBE}(I(M, cred.nym||cred.attr), cred')$ for some $cred' \neq cred$, the value obtained is computationally indistinguishable from a random value (i.e., we assume that decryption with the wrong key is a random oracle).

The hidden credentials system is secure against an adaptive chosen ciphertext attack where an attacker can obtain an unlimited number of other arbitrary credentials [86]. The hidden credentials are also unforgeable [86]. We now give a simple example of how Alice can access Bob's resource using a hidden credential. Suppose Bob's resource M can only be accessed by a student. Alice has a student credential *cred*, i.e., $cred.nym = Alice$ and $cred.attr = student$. To access M, Alice sends her username *Alice* to Bob. Bob responds with $E_{IBE}(M, Alice||student)$. Alice uses her credential *cred* to decrypt $E_{IBE}(M, Alice||student)$ and obtains M. Note that, Bob does not learn if Alice possesses a student credential from the above interaction.

4.4.2 Protocol Phases

There are two primary phases in our protocols: i) a credential hiding phase and ii) a blinded policy evaluation phase. During the credential hiding phase, Alice and Bob engage in a protocol that in some way (to be specified later) hides which credentials that Bob's policy requires. During the blinded policy evaluation: if Alice satisfies Bob's policy then she learns the requested message, and learns nothing about the message if she does not satisfy Bob's policy. We now describe each phase in more detail:

- *Credential Hiding Phase:* Suppose Bob's policy contains n attributes $attr_1$, $\ldots$, $attr_n$. At the end of this phase Alice has a set of values $\ell_1, \ldots, \ell_n$ (i.e., one for each attribute), where $\ell_i \in \{r_i[0], r_i[1]\}$, which are values generated by Bob. These values will either be encryption keys or seeds for a pseudo-random generator that can produce such keys. The value of ℓ_i is subject to the following constraints:

  1. $\ell_i = r_i[1]$ only if Alice has a credential *cred* such that *cred.attr* $=$ $attr_i$ (i.e., if Alice does not possess attribute $attr_i$ she cannot learn the value $r_i[1]$). Otherwise Alice gets $r_i[0]$.
  2. A computationally-bounded Alice learns nothing about the value $\{r_i[0], r_i[1]\} - \{\ell_i\}$.

- *Blinded Policy Evaluation Phase:* Given the ℓ values from the previous phase, Alice and Bob engage in a protocol that allows Alice to learn message M if she satisfies Bob's policy. His policy is represented by a Boolean function $p : \{0,1\}^n \rightarrow \{0,1\}$ (i.e., it maps n values, which correspond to which attributes Alice has, to a binary value that corresponds to whether or not Alice satisfies Bob's policy or not). To formalize the definition of this phase, suppose that, after the previous phase, Alice's values are $r_1[x_1], r_2[x_2], \ldots, r_n[x_n]$, where $x_i \in \{0,1\}$, then Alice will receive M if and only if $p(x_1, x_2, \ldots, x_n) = 1$.

4.4.3 Credential Hiding Phase

In this Section, we introduce three protocols for the credential hiding phase. Figure 4.1 defines the input and output for this phase. There is an inherent security/communication complexity tradeoff for these protocols. We denote the number of attributes in Bob's policy (it may be an upper bound) by n, the number of credentials Alice is willing to use (it may be an upper bound) by m, and a security parameter by ρ. The protocols can be summarized as follows:

1. *Protocol 1*: In this protocol it is assumed that Bob is willing to reveal to Alice a superset of the attributes in his policy (he chooses this superset and can make it large enough to achieve a "hiding in a crowd" effect that suits him). While this is not acceptable for all applications, there are many cases where Alice could guess with high probability the set of attributes in Bob's policy before the protocol, and in such cases this protocol may be acceptable to Bob. The example given below is a scenario where Bob is willing to reveal a superset of attributes in his policy. The communication complexity of this protocol is $O(\rho n)$ and it requires 3 message exchanges.

2. *Protocol 2*: Unlike Protocol 1, this protocol does not assume that Bob is willing to reveal a superset of the attributes in his policy. In this protocol, Bob learns the value m and Alice learns: (i) the value n and (ii) the number of attributes in Bob's policy that she satisfies (she does not know which of her credentials are responsible for this). This protocol requires $O(\rho mn)$ communication and 5 message exchanges.

3. *Protocol 3*: This protocol is similar to Protocol 2, but Alice does not even learn how many attributes she satisfies in Bob's policy. This protocol requires $O(\rho^2 mn)$ communication and 5 message exchanges.

Example 6 An online auto insurance company gives a special promotion to those who are married and have good credit history. The policy for this special promotion is $married \wedge good\_credit$. The insurance company may publish a superset of attributes that the policies may contain such as age, gender, marital status, credit history, number of accidents in the last three years, state of residence, etc. The actual promotion policy contains only a small subset of the attributes. The auto insurance company may treat the superset of the attributes as non-sensitive information, but treat the attributes used in the policy as well as the policy structure as private information.

Input: Bob has a policy P relative to set of attributes $attr_1, \ldots, attr_n$, and for each attribute $attr_i$ Bob has two random values $r_i[0]$ and $r_i[1]$. Alice has a set of credentials $cred_1, \ldots, cred_m$.
Output: Alice learns a value ℓ_i for each attribute $attr_i$, where $\ell_i = r_i[1]$ if and only if there exist a $cred_j$, $1 \leq j \leq m$, in her credentials set such that $cred_j.attr = attr_i$, and is $r_i[0]$ otherwise. Recall that a crucial element of this protocol is that Alice learns exactly one value for each attribute.

Figure 4.1. Input and Output of Credential Hiding Phase

Protocol 1

This protocol (see Figure 4.2) assumes that Bob is willing to reveal to Alice a superset of the attributes in his policy. While this is not always reasonable, there are situations where this is acceptable. This is the most efficient of the protocols, but also reveals the most information about the attributes in Bob's policy.

Input/Output: See Figure 4.1.

1. For each attribute $attr_i$: Bob generates a random key k_i as well as information C_i that reveals to Alice what credential she needs to satisfy $attr_i$. Bob then generates an encryption $\alpha_i = E_{IBE}(k_i, nym||attr_i)$. He sends the following information to Alice: $(\alpha_1, C_1), \ldots, (\alpha_n, C_n)$.
2. For each ordered pair (α_j, C_j), Alice generates a value b_j which is 1 if she possesses a credential $cred_p$ that satisfies C_j and is 0 otherwise. If $b_j = 1$ she computes k_j which is $D_{IBE}(\alpha_j, cred_p)$.
3. For each attribute $attr_i$, Alice and Bob engage in a chosen 1-out-of-2 OT protocol where Bob's input is the list $\{r_i[0], Enc(r_i[1], k_i)\}$ and Alice's input is b_i.
4. If b_i is 1, then Alice decrypts $Enc(r_i[1], k_i)$ with k_i (she computed this in Step 2) and sets the result as her output, and otherwise she sets her output to be $r_i[0]$.

Figure 4.2. Credential Hiding Phase Protocol 1

Complexity Analysis: For each attribute the protocol requires a single encryption to be sent between Alice and Bob as well as the descriptions C_i (which we assume are of size $O(1)$), and a single chosen 1-out-of-2 OT. Thus, the communication complexity is $O(\rho n)$. The OT can be started during the first round and thus this protocol requires only 3 message exchanges.

Intuition: The intuition of this protocol is that if Alice does not possess $cred_i$, then she cannot obtain k_i. Furthermore, without k_i the value $Enc(r_i[1], k_i)$ reveals nothing about $r_i[1]$ (in a computational sense). And since OT is used she can get at most one of the values.

Protocol 2

In this protocol (see Figure 4.3) Bob does not reveal to Alice a superset of the attributes in his policy, but Alice learns how many (but not which) attributes she satisfies in Bob's policy. Clearly, if Bob pads his list with superfluous credentials as in Protocol 1, this reveals less information to Alice.

Input/Output: See Figure 4.1.

1. For each attribute $attr_i$, Bob generates two keys $k_i[0]$ and $k_i[1]$ and a public marker q. He also computes $\alpha_i = E_{IBE}(k_i[0], nym||attr_i)$. He then sends to Alice $\alpha_1, \ldots, \alpha_n$ along with q.
2. Alice generates a semantically-secure homomorphic encryption system E_A. Then for each value α_i and for each of her credentials $cred_j$, she creates a value $\beta_{i,j} = D_{IBE}(\alpha_i, cred_j)$. Alice and Bob then engage in set intersection protocol where Bob inputs $k_i[0]$ and Alice inputs $\{\beta_{i,1}, \ldots, \beta_{i,m}\}$ and where Bob learns $E_A(\gamma_i)$ where γ_i is 0 if Bob's element is in Alice's set and is a random value otherwise. Bob then computes $\delta_i = E_A(\gamma_i) * E_A(k_i[1]) = E_A(\gamma_i + k_i[1])$; he then forms ordered pairs $(\delta_i, Enc(q, k_i[1]))$. Bob randomly permutes these pairs and sends them to Alice.
3. For each value $(\delta_j, Enc(q, k_j[1]))$, Alice computes $\kappa_j = D_A(\delta_j)$, and then computes $Dec(Enc((q, k_j[1])), \kappa_j)$. If this value is q, then she stores κ_j (which is $k_j[1]$) and sets b_j to 1, and otherwise she sets b_j to 0.
4. For each attribute $attr_i$ (note that these are permuted), Alice and Bob engage in a chosen 1-out-of-2 OT protocol where Bob's input is the list $\{r_i[0], Enc(r_i[1], k_i[1])\}$ and Alice's input is b_i. Note: that these values are permuted by Bob above, but Alice does not need to know which value corresponds to which initial attribute.
5. If b_i is 1, then Alice decrypts $Enc(r_i[1], k_i[1])$ with $k_i[1]$ (she computed this in Step 3) and sets the result as her output, otherwise she sets her output to be $r_i[0]$.

Figure 4.3. Credential Hiding Phase Protocol 2

Complexity Analysis: It is clear from the above that, for each attribute, $O(m)$ modular exponentiations are required (in the form of homomorphic/identity-based encryptions and decryptions). Thus there are $O(mn)$ such operations, and thus the system requires $O(\rho mn)$ communication. Furthermore, the OT in Step 4 can be started in Step 3, and thus this protocol requires 5 message exchanges.

Intuition: That Alice cannot retrieve both $r_i[0]$ and $r_i[1]$ follows from the fact that OT is being used. To see why the value $r_i[1]$ cannot be obtained if she does not possess the credential, note that to obtain this value she must have $k_i[1]$. She can learn this value only when one of the values computed in Step 2, is $k_i[0]$, which is the case only when Alice has the credential (with all but negligible probability).

Protocol 3

This protocol (see Figure 4.4) protects which attributes are in Bob's policy more than the previous protocols in that Alice does not learn how many credentials she satisfies in Bob's policy. As stated earlier this protocol requires more communication and computation than the previous protocols. It also uses Scrambled Circuit Evaluation extensively.

Input/Output: See Figure 4.1.

1. For each attribute $attr_i$, Bob generates two keys $k_i[0]$ and $k_i[1]$. He also computes $\alpha_i = E_{IBE}(k_i[0], nym||attr_i)$. He then sends to Alice $\alpha_1, \ldots, \alpha_n$.
2. Alice generates a semantically-secure homomorphic encryption system E_A. Then for each value α_i and for each of her credentials $cred_j$, she creates a value $\beta_{i,j} = D_{IBE}(\alpha_i, cred_j)$. Alice and Bob then engage in set intersection protocol where Bob inputs $k_i[0]$ and Alice inputs $\{\beta_{i,1}, \ldots, \beta_{i,m}\}$ and where Bob learns $E_A(\gamma_i)$ where γ_i is 0 if his element is in Alice's set and is a random value otherwise. Bob then computes $\delta_i = E_A(\gamma_i) * E_A(k_i[1]) = E_A(\gamma_i + k_i[1])$. Bob sends $\delta_1, \ldots, \delta_n$ values to Alice.
3. For each, value δ_j, Alice computes $\eta_j = D_A(\delta_j)$. Alice and Bob engage in a Secure Circuit Evaluation with Bob as the generator and Alice as the evaluator. The circuit that Bob creates is a circuit for an equality test (which requires $O(\ell)$ gates for ℓ-bit values). Bob's input into the circuit is $k_j[1]$ and Alice's input is η_j. If the values match the circuit outputs $r_j[1]$, otherwise it outputs $r_j[0]$.
4. Alice sets her output to the output of the circuit.

Figure 4.4. Credential Hiding Phase Protocol 3

Complexity Analysis: There are n circuits defined above, each of which compares m values with ρ bits. Thus there will be $O(\rho mn)$ modular exponentiations and $O(\rho^2 mn)$ communication. This can be reduced slightly by using a one-way function on the values in Step 4 before engaging in the circuit. The protocol requires 5 message exchanges.

Intuition: That Alice cannot retrieve both $r_i[0]$ and $r_i[1]$ follows from the fact that, in SCE, the evaluator learns only one encoding per wire (which includes the output wire). To see why she can learn the value $r_i[1]$ only if she possesses the credential, observe that in order to learn this value Alice must be able to (in Step 2) select m values such that one of them is $k_i[0]$, but this is intractable unless she has a credential that decrypts the value $k_i[0]$.

4.4.4 Blinded Policy Evaluation Phase

In this Section we outline a protocol for blinded policy evaluation. Notice that after the previous phase the values could be used as input wires into a scrambled circuit (as Alice knows only one value and Bob knows the encodings). Thus the strategy will be to use an oblivious circuit. In Figure 4.5 we formally define this phase.

We now give several examples of useful oblivious circuits for this problem:

1. A binary tree of oblivious gates could be used to compute several common types of policies, including: conjunction (does Alice have all of the attributes?), disjunction (does Alice have at least one of the attributes?), conjunction/disjunction of a subset (does Alice have all(one of) of a subset of the attributes), and other policies. Note that the size of this circuit is $O(\rho n)$.

2. An addition circuit followed by a comparison circuit, would allow for computation of a threshold based function (i.e., "does Alice have at least 4 of the credentials"). This could easily be modified to support policies that require a

Input: Bob has a policy $p : \{0,1\}^n \rightarrow \{0,1\}$, several pairs of values $\{r_1[0], r_1[1]\}, \ldots, \{r_n[0], r_n[1]\}$, and a message M. Alice has n values $\ell_1, \ldots, \ell_n$ where $\ell_i \in \{r_i[0], r_i[1]\}$.
Output: Alice learns M if and only if $P(\ell_1 \overset{?}{=} r_1[1], \ldots, \ell_n \overset{?}{=} r_n[1]) = 1$, and learns nothing otherwise.
Steps:
1. Bob constructs a circuit C that computes his policy (several "useful circuits" are described below) that uses the r_i values as inputs and that has an output wire with two encodings: k and some other random value. He sends the encodings of the circuit's gates to Alice (note that she already has input values) along with $Enc(M, k)$.
2. Alice evaluates the circuit and then tries to decrypt the message with the value she computes for the output wire of the circuit.

Figure 4.5. Protocol for Blinded Policy Evaluation

threshold number of attributes for a subset of the attributes. Note that the size of the circuit is $O(\rho n)$.

3. By having one instance of each of the above, combined with a single oblivious gate it is possible to have policies that are combinations of: conjunction, disjunction, and threshold based. Note that the circuit size is still $O(\rho n)$.

4. More complex policies can be represented by combining several of the above defined oblivious circuits together and then connecting them with other circuits. Of course, when this is done the structure of the policy is revealed slightly (as the topography of the circuit is revealed). However, the structure is at a high level, and the individual pieces of the policy are hidden (thus the evaluator does not learn things like whether or not the binary tree being used is for conjunction/disjunction, the thresholds that values are being compared to, and which attributes are being used).

Clearly a large class of useful policies can be used with this technique even without universal circuits or arbitrary circuits, but if this level of expressiveness is required then these more expensive techniques can be used.

4.4.5 Security Proofs

The proof consists of two parts. The first is showing that the composition of the Credential Hiding Phase and the Blinded Policy Evaluation Phase is secure. The next part is to show that the protocols described are secure and do not leak additional information.

Proof of Composition

We now show that the protocol is secure according to the definition presented in Section 4.1.2.

Theorem 4.4.1 *Given trusted oracles CHP and BPE where CHP provides the Credential Hiding Phase functionality (see Figure 4.1) and BPE provides the Blinded Policy Evaluation functionality (see Figure 4.5), the protocol is secure (in as strong of a model as the weakest of CHP and BPE).*

Proof: The *CHP* oracle requires that Bob input n triples into the system $(attr_i, r_i[0], r_i[1])$, that Alice learns one of Bob's r_i values per tuple, and that she learns the $r_i[1]$ only if she has a credential satisfying $attr_i$. Now *BPE* requires that Bob inputs pairs of the form $(r_i[0], r_i[1])$ and a policy P, that Alice inputs a single value from the pair $r_i[x_i]$, and that she gets M iff she $P(x_1, \ldots, x_m) = 1$. In order to make multiple distinct probes at *BPE* she must be able to learn two values from the same attribute pair, which is not possible since the *CHP* oracle allows Alice to learn at most one value (the values can be chosen to be from a large enough space as to prevent guessing). Also, she obtains the message only when she has a set of attributes that matches the policy. Up to this point we have been informal and have not discussed the effect of the information leaked by our protocols (e.g., an upper bound on the number of attributes in the policy). As this is a fairly simple extension of the above proof we do not give all of the details here, but this information does not give the adversary any significant advantage in breaking the above system. $\square$

Proof of Credential Hiding Phase

We must show that the protocols satisfy four properties: i) that Alice gets at most one of the r_i values per attribute, ii) that she only obtains the value if she possess a credential $cred_i$ for the attribute, iii) that Bob gains only the previously mentioned information from the protocol, and iv) that Alice gains only the previously mentioned information from the protocol. We do not give formal proofs for (iii) and (iv) as these proofs are natural consequences of the properties of encryption, semantic-security, secure implementations of OT, set intersection, and scrambled circuit evaluation(SCE).

Property i: For protocols 1 and 2, this is obvious because these values are sent with OT, which implies that she obtains only one value. Protocol 3 has this property since it uses SCE, and a condition of SCE is that the evaluator learns only one encoding per wire.

Property ii: In Protocol 1, Alice learns the value only when she has the credential since the message is encrypted with k_i. In Protocol 2, she needs to learn $k_i[1]$ in order to get the message, but she obtains $k_i[1]$ only when she obtains $k_i[0]$. She obtains this value only when she has a credential for the attribute. Protocol 3 satisfies this property by the correct construction of a circuit and the argument for Protocol 2.

Property iii: Given secure implementations of OT, set intersection, and SCE in the malicious model, it is trivial to see that the protocols are nothing more than compositions of these protocols. However, the protocols leak information that was previously specified (such as an upper bound on the number of credentials in the policy).

Property iv: Similar to the previous property.

Proof of Blinded Policy Evaluation

As the protocol for Blinded Policy Evaluation is just SCE all that needs to be shown is that Alice learns the message only when she has a set of values that correspond to satisfying the policy. As long as Alice knows only one value per input wire then she cannot learn anything other than at most one value for the output. Since Bob generates the circuits to match his policy, the correctness is guaranteed.

4.5 Protocols for Credit Checking

In this Section we introduce the protocols for credit checking. We give a preliminary protocol in Section 4.5.1. We discuss the security of the protocol in Section 4.5.2. Finally, we discuss several extensions in Sections 4.5.3-4.5.7.

4.5.1 A Preliminary Protocol

In this Section we introduce a protocol for achieving privacy-preserving credit checking. This is not the final protocol, but rather it should be viewed as a "warmup" for the better protocols that come later on. Furthermore, we use a modular approach to describe the protocol, but the round efficiency can be greatly improved by combining many of the phases.

Protocol Description

Phase 0: Setup: For each borrower, the CRA has a credit report represented by attributes $a_1, \ldots, a_m$. When Bob registers with the CRA, the CRA and Bob establish a shared encryption key (call it k).

Phase 1: Loan Request: When Bob attempts to establish a loan with Linda, she contacts the CRA to obtain information about Bob's credit report. When the CRA receives a request for a credit report, the CRA generates two keys (for a symmetric encryption scheme) for each attribute (note that these values can be pre-computed).

Let the pair for attribute a_i be denoted by (e_i^0, e_i^1). The CRA sends to the Lender the following two things: i) All of the key values $e_1^0, \ldots, e_m^0, e_1^1, \ldots, e_m^1$, and ii) the key values encrypted with the key k (i.e., $Enc((e_1^{sat(a_1)}, \ldots, e_m^{sat(a_m)}), k)$). The Lender stores the first part of this message, but forwards the second part to the Borrower who decrypts the value and stores the result. In what follows, the goal is to build a circuit that determines if the Borrower qualifies for the loan. Thus we need to address how to obtain the circuit's input, how to build a circuit, and how to obtain the results from the circuit.

Phase 2: Attribute Input: In this phase Linda chooses two keys (for a symmetric encryption function) for each attribute, and Bob learns one of the keys if he satisfies the corresponding attribute but learns the other key if he does not satisfy the attribute. Let the keys for attribute a_i be denoted s_i^0 and s_i^1, where the borrower will learn $s_i^{sat(a_i)}$. For each attribute a_i, Linda computes an ordered pair $(Enc(s_i^0, e_i^0), Enc(s_i^1, e_i^1))$. She sends all of these pairs to Bob. For each attribute a_i, Bob computes $Dec(Enc(s_i^{sat(a_i)}, e_i^{sat(a_i)}), d_i^{sat(a_i)})$. After this step Bob has only the keys that correspond to his attributes. Note that it is possible for Linda and Bob to skip the previous step by using the values (e_i^0, e_i^1) as Linda's output and $e_i^{sat(a_i)}$ as Bob's output for the phase. However, this does not allow Linda to precompute circuits for loan qualification.

Phase 3: Determining Satisfiability of Criterion: For each criterion C, we need to be able to compute $sat(C)$. A criterion C has a certain set of attributes, and for each attribute there are two possible keys (one corresponding to Bob satisfying the attribute and another to Bob not satisfying it), and his input to the circuit will be his corresponding keys for the attributes. In order to be able to use a circuit on the output of this satisfiability computation the output of this part should be one of two keys (which Linda chooses), where one corresponds to a borrower that satisfies the criterion and the other corresponds to a borrower that does not satisfy the criterion. If the criterion is a single attribute, then this is trivially done by using the attribute's input key as the output key. If the criterion is a comparison, then a standard circuit for comparison can be used. Thus either type of criterion can be implemented in communication proportional to the number of attributes in the criterion.

Phase 4: Result Combination: After the previous phase, for each criterion Linda has two keys and Bob has one of these keys. During this phase Linda and Bob evaluate a circuit that determines if he qualifies for the loan. Recall our assumption that Bob qualifies if he satisfies at least t criteria, in other words Bob qualifies if $\sum_{i=1}^{n} sat(c_i) > t$ for some $t \leq n$. This can easily be done with the addition circuit, followed by a comparison circuit with value t. Note that the size of this circuit is of size proportional to number of criteria.

Phase 5: Obtaining Result: After the previous phase, Bob has one of two keys k_0 or k_1 (both of which were generated by Linda). Key k_1 corresponds to Bob qualifying for the loan, and key k_0 corresponds to Bob not qualifying for the loan. The question now becomes how does Bob prove he qualified for the loan to Linda. Bob proves this to Linda by sending her the key that he receives (recall that we assume the honest-but-curious model in this initial protocol). When Linda receives k_1 from Bob, she is convinced that he qualified for the loan, and if instead she receives k_0 then she denies him the loan.

Summary of the Protocol

We now state the above protocol in more concise terms, leaving out the details but describing the information flow:

1. To use the privacy-preserving credit check, Bob registers with the CRA and sets up an account for this service, the CRA and Bob establish a private key (call it k).

2. When Bob requests a loan from Linda, she contacts the CRA with a request to run a protocol that determines for her whether Bob's credit report satisfies her criteria (without revealing the report itself to her).

3. The CRA sends Linda a set of encryption keys; there are two keys for each attribute (one corresponding to each possible value of the attribute). The CRA also sends a single encryption key for each attribute (the one that corresponds to Bob's actual attribute) encrypted with the value k.

4. Linda builds a scrambled circuit that determines if Bob qualifies for a loan. To obtain the input for the circuit, she encrypts the encodings with the encryption functions that she received from the CRA. She sends Bob the following items: the circuit, a set of messages from which he can obtain the input wire encodings of the circuit, and the message from the CRA with the decryption keys.

5. Bob decrypts the decryption keys and obtains the inputs to the circuit using these keys. He then evaluates the circuit and upon finding the output, sends the result to Linda.

6. Linda tells Bob whether or not he qualifies for the loan.

The communication complexity of the above is proportional to the number of attributes, and requires one round of messages between Linda and the CRA and another between Bob and Linda. The CRA must generate the key pairs for each attribute, but this is not expensive because it is just random number generation. The CRA must also encrypt Bob's encryption keys, but this can be done using a fast symmetric key encryption system (such as AES). Linda needs to do two symmetric encryption operations for each attribute and needs to generate a circuit for the credit check, but the latter can be pre-computed. Bob needs to perform a single symmetric key operation for each attribute, and then needs to evaluate a circuit (where the circuit has size proportional to the number of attributes), but since this is in the passive model these operations can be a symmetric key system. The correctness of the protocols trivially follows from the correctness of secure circuit evaluation.

4.5.2 Security of Protocols

We now discuss the security of the above protocol.

Borrower: During Phase 2, the borrower has only one key per attribute and thus he can obtain at most one input for the circuit. Thus the borrower can find at most one value for the output of the circuit (assuming he is computationally bounded).

Lender: This design goals for the lender are trivially satisfied in the passive model as the lender learns only a single value after the computation of the circuit, and because the message from the CRA is encrypted with a key unknown to Linda.

CRA: The CRA sees only a request from Linda about Bob.

4.5.3 Extension: Weighted Threshold Based Policies

In the protocol in Section 4.5.1, Bob qualifies for a loan if he satisfies at least t out of n criteria (where t is a private parameter defined by Linda). This can be generalized to add more flexibility to the system. For example, some criteria may be more important than other criteria. In the case where the criteria are globally defined (by and for all lenders) some of the criteria may not apply to Linda. In this Section we propose a weighted threshold based policy. Each criterion c_i is assigned a weight w_i, where $w_i \in \{0, \ldots, N-1\}$. Bob qualifies if $\sum_{i=1}^{n}(sat(c_i)w_i) > t$ for some threshold t (again t is privately defined by Linda). The techniques in Section 4.5.1 can be used, however between phases 3 and 4 another step must then be added to replace $sat(c_i)$ with $w_i sat(c_i)$. Essentially, each criterion value must be expanded to a $(\log N)$-bit scrambled value. This can easily be done with a 2-ary gate having $(\log N)$ outputs, and size $O(\log N)$. The circuit is also changed to add the larger values. Note that the circuit size is now $O(m + n \log N)$ where m is the number of attributes and n is the number of criteria.

4.5.4 Extension: Combinatorial Circuit Based Policies

The previous notion of a weighted policy allows the definition of many policy types, but it does not allow policies of the following forms:

1. "(At least 3 out of these specific 4 criteria) and (At least 2 out of these specific 5 criteria)".
2. "Criterion A or (Criterion B and Criterion C), but not (Criterion A and Criterion B).

Thus if the policy has a complex structure, then there may not be a way to represent it with a weighted threshold system, and even if there is such a representation, it may not be intuitive. In this Section, we propose a more generic structure that is more powerful and more intuitive. For this representation the lender defines several "super-criteria", which are expressible as threshold policies, weighted threshold policies, or other combinatorial based circuit policies. The lender defines a set of these super-criteria and then combines the results using a binary tree of oblivious gates. Clearly, anything that can be expressed with threshold policies or weighted threshold policies can be expressed with this mechanism, and the examples given above can easily be expressed with this representation. This reveals the general structure of the policy but none of its specifics – actual values of thresholds, cutoffs, etc. However, to hide the structure, padding the circuit with "dummy" entries can hide further the structure of the circuit.

4.5.5 Extension: General Policies or Criteria

While the combinatorial circuit-based policies described in the previous Section allow the lender to represent a large class of policies, it is still limited. In addition, complicated criteria composed of multiple attributes cannot be represented. To handle such requirements one could extend the techniques to represent an arbitrary policy (by using an n input gate that represents the policy). However, such an extension would require exponential communication/computation. Another option is to use a universal circuit.

4.5.6 Extension: Malicious Adversaries

In our previous protocols we assumed that the behavior of the parties is passive or honest-but-curious (i.e., that the parties will follow the protocols, but will try to glean additional information). We postpone discussion of malicious CRAs until Section 4.5.7, and thus in this Section we assume that the CRA is honest (as it has very little incentive to behave otherwise). However, in this Section we explore what happens when Linda or Bob deviates from the protocol.

- As a borrower, there is little Bob can do to deviate from the protocol. The only type of attack we worry about is Bob falsely qualifying for a loan, and since his only messages are his loan request and the message which states whether or not he obtained the loan (which he cannot forge), a malicious Bob has no extra advantage.
- Linda violates Bob's privacy if she learns additional information about Bob's credit report. She can do some things to try to probe Bob' credit, and there are essentially three ways in which she can do this: i) abort the protocol after receiving the results, ii) having the output gate output more than two things, iii) creating a malformed circuit. We now examine each of these in more detail (note that this solution does not require the gates to be some form of encryption that can be verified with Zero-Knowledge proofs):

  1. Linda could create a circuit that computes something other than whether or not Bob qualifies for a loan, and then she could abort the protocol after receiving Bob's response. This "abort" could take the form of a contrived network failure, and Bob's only choice would be to run the protocols again with Linda. To avoid this we propose a modification of Phase 5 of the

protocol. Instead of revealing a random string, the circuit reveals one of two random keys. As part of the circuit Linda also sends a message, which is an encryption with the "qualify" key of a signed message stating that Bob qualifies for the loan. Now, Linda cannot do the above abort attack, because Bob can resend the signed message.

2. Linda could make the output of the gate have more than two outputs, which would reveal additional information to her about Bob's credit report. However, this is already handled by the solution offered for the previous problem, assuming that it is hard to generate two different keys that encrypt to the same value for two different signatures.

3. Linda could send a circuit that is malformed (i.e., some inputs will lead to an output, but other sets of inputs lead to Bob not being able to compute the output by not being able to decrypt any message at a gate). If Bob reports a problem, then a malicious Linda has more information since there are now two fail states. To avoid this Bob, sends the same message if there is an error in the circuit as if he gets to the output gate but does not qualify.

4.5.7 Extension: Multiple CRAs

The assumption that there is a single CRA is not realistic because in practice there are multiple CRAs. If the information is used in independent ways, then this can be handled by a trivial extension to our protocols. Thus we consider the case where the lender uses multiple CRAs and wants to make sure that the borrower satisfies the criteria at all of the CRAs (which may have conflicting information). When there are multiple sources for the same information, a lender would like to add "defense in depth", and we propose a conflict resolution strategy that resolves conflicts with the safest possible approach. Essentially the circuit computes the criterion for each of the CRAs, and then combines them into a single value. If the criterion is positive (i.e., it helps Bob get the loan) then the lender would want to make sure that all of the CRAs information states that Bob satisfies that criteria, which can easily be done with a set of AND gates. If the criterion is negative (i.e., it hurts Bob's chances of getting the loan) then the safest approach that Linda could take is that Bob has the property in question if one or more of the CRAs claim he has that property, and this can easily be computed with a set of OR gates. Clearly, either of these cases can be done with a binary tree of oblivious gates.

In the single CRA case, a single malicious CRA that colludes with one or more borrowers to probe a lender's policy is unavoidable. However, when multiple CRAs are used and the policy is to use the safest possible approach (as outlined above), then unless all of the CRAs are corrupt, the colluding parties are limited in what they learn about the lender's policy. The probing of the malicious CRAs is limited to credit reports that are worse than the borrower's credit report with the non-colluding CRAs.

4.6 Summary

In this Chapter, we introduced a variation on standard secure function evaluation called Secure Verifiable Oblivious Function Evaluation. There are several things that differentiate this work from traditional SFE including: i) the function that is to be computed is private to one party, ii) the result of the function is typically revealed to only one party (and this does not need to be the party with the function), and iii) one party's inputs must be verified by a third party that is not online during the protocols. We introduced general techniques for Secure Verifiable Oblivious Function Evaluation, and analyzed their applicability to two applications: trust negotiation and credit checking. While we only presented this work for these two domains, they have much broader applicability and in fact work for any situation where there is a repository of public and private information about individuals, that is subsequently used for making decisions that impact the individuals (a credit rating agency is but one example of such a repository). Other repositories contain information collected and maintained for different purposes than credit ratings, e.g., information about peoples' shopping habits and preferences, level of education, income, number and age of their children, history of contributions and donations to charities, etc. The information in these repositories is typically collected without the active and knowing participation of the individuals concerned, and the repositories often contain wrong information that the individual never gets a chance to correct. Moreover, these repositories are often used for purposes the individual would not approve of, such as junk mail, spam, and more nefarious purposes such as identity theft. A privacy-respecting repository of the kind we envision may, because its reputation and business model are based on offering privacy to its customers, cause individuals to actually contribute additional (and accurate) information to the repository, thereby increasing its value over the competition's. Such a repository may have information (backed with proof or certification) that John Doe is fluent in French, has a degree in Mechanical Engineering, has extensive experience in the analysis of seismic data, all in addition to the usual financial and personal information stored in today's CRAs. The lender-like entities who use such a repository would inquire about a specific individual only with the individual's permission (e.g., when he applies for a loan, a security clearance, a permission to buy restricted merchandise) would benefit not only from the more accurate information in the repository, but also from the fact that no one (including the repository) will know the precise nature of their credit-like inquiries about an individual; this is especially important when screening a candidate for a security-sensitive position, for access to restricted online material, for eligibility to a special discount, or any other situation where there is fear that a malevolent individual could "game" the screening system if such an individual knew the precise screening criteria and algorithm.

5 SERVICES FOR LOCATION-AWARE DEVICES

5.1 Introduction

Suppose Bob is visiting an unfamiliar area, and that Bob has a mobile device which provides him with location information. Furthermore, Bob would like to use this location information to determine if a certain type of store (coffee shop, rare bookstore, etc.) is near a route that he will travel on, but for privacy reasons Bob does not want to reveal his route. Suppose Alice has gathered a large amount of information and that she provides a subscription-based service to access this information. However, Alice cannot simply reveal all of her information to Bob since this reveals all of Alice's proprietary information about certain topics, which is unacceptable because this is how Alice generates revenue (not to mention that Alice's database may be too voluminous to send to Bob). Thus the problem becomes can Bob learn if there is a specific item of interest near his route, without revealing his route and without requiring Alice to reveal all of her information.

A similar application of this technology is to help entities with restraining orders placed against them to avoid the person that they are ordered to avoid. Suppose Alice has a restraining order on Bob (perhaps she has caught him being malicious in one too many protocols) that states that he must stay at least 1000 yards from her. Clearly, Alice wants to stay away from Bob, and Bob would also like to stay away from Alice. Furthermore, Alice and Bob have the right to privacy, and it would be potentially dangerous for Alice to tell Bob where she will be at all times during the day. Thus, Alice and Bob would like to engage in a protocol that tells them if their routes are safe (i.e., does not get too close to the other party's route) without revealing their routes to each other.

There are also several natural applications for cooperating but mutually distrusting counties or military organizations (so-called "uneasy allies"). Suppose Alice and Bob are two governments who are temporarily cooperating to perform a humanitarian relief (or perhaps a military) operation. Bob has an object (vehicle, convoy, or airplane) that is moving in a space where Alice has certain objects that she wants to hide from Bob. Although Alice and Bob are cooperating, they do not fully trust each other; furthermore the less information is disseminated, the smaller the risk that this information will be leaked by an untrustworthy partner (or by a crooked insider employed by an otherwise trustworthy partner). A more extreme example is when both Alice and Bob have moving objects that they want to prevent from getting too close to each other in order to reduce the likelihood of an accidental collision or of a "friendly fire" accident.

Another related problem is to help Bob find the nearest object of interest to his route (rather than learning if his route is "close" to a set of objects). This is a nearest neighbour problem. Applications for such a protocol include: i) determining the nearest coffee shop to a route, and ii) determining which of Bob's friend's routes is closest to his to determine if a meeting is possible (or avoidable depending on the friend).

The above problems are all instances of computational geometry problems where the operation being computed is proximity between objects. These objects can take various forms: points, points moving in space (defined by parametric equations), and line segments. Furthermore, the distance between these objects must be computed in a secure fashion. This computation should be done in a way that does not reveal anything other than the result (or what can be computed from the result, as this is unavoidable in any such protocol). We assume that the inputs (i.e., point coordinates, coefficients of equations, velocities, etc) are integers – so there is an implied integer "grid"; we also assume that all coordinates are non-negative. Furthermore, we give protocols for two dimensions, however our protocols can easily be extended to higher dimensions. In this Chapter we introduce protocols to compute the following quantities:

1. The distance between a segment and point.
2. The distance between two points moving with constant velocity, described by parametric equations.
3. The distance between two line segments.

One of the drawbacks of the approach outlined above is that the client is at the mercy of the sever (i.e., if the server does not want to engage in such a protocol then such techniques are of no use to the client). Because of this reason we also explore various “self-protecting” privacy solutions. Our focus is on perturbation based techniques for location based queries. Thus the client will obfuscate his current location and send this query to the server. The server will then respond to this obfuscated query. Of course such a solution leads to inaccuracies in the answer of the query, but this is a tradeoff that the client can choose. For example, there are cases where privacy is less of a concern than efficiency (i.e., if the client is low on gas and wants to know the nearest gas station). This approach has the advantage that the client can use this approach even when the server is not cooperating. It is worth noting that in many cases the server will have out of band information about the approximate location of the client; for example when the device is a cell phone and the server is the cell phone service provider, then the server knows the approximate location of the cell phone based upon which cell that it is in. In this situation, these perturbation based techniques are even more attractive as revealing to the server one’s approximate location does not reveal substantially more information than what the server already knows.

The rest of this Chapter is organized as follows. In Section 5.2 we define the notations that are used in this Chapter. In Section 5.3 we introduce secure protocols for computing distance functions. In Section 5.4 we introduce several perturbation based techniques. Finally, we summarize our results in Section 5.5.

5.2 Notations and Definitions

The following is a list of notations that are used within this Chapter.

1. We use $<>$ to represent vectors.
2. To represent a point in two dimensional space we use the following notation: $P(x, y)$.
3. To represent a line in two dimensions we use $L(A, B, C)$, where the line is described by the equation: $Ax + By + C = 0$.
4. To represent a line segment we use the notation $S(P_1(x_1, y_1), P_2(x_2, y_2))$ for the segment between the two points P_1 and P_2.
5. To represent motion of a point along a line with constant velocity we use parametric equations: $x(t) = (v_x)t + x_0$ and $y(t) = (v_y)t + y_0$ to represent the x and y coordinates of the moving point.

5.3 Secure Protocols

In this Section, we describe secure protocols for distance functions. Atallah and Du have studied various secure protocols for distance functions [105]. This work focused on solving three problems: i) the inclusion of a point in a polygon, ii) the intersection of polygons, and iii) the distance between points; solutions were given for all of these problems. This Chapter focuses on distance protocols rather than on intersection protocols, but some of the protocols overlap. The two cases where this Chapter's protocols overlap the work of [105] are: i) the computation of the distance between points and ii) the determination if two line segments intersect. Instead of placing these protocols in a building block Section we place them with the protocols that use them, however they are clearly marked as previous work.

The rest of this Section is organized as follows. In Section 5.3.1 we discuss the difficulties of evaluating distance functions with general SMC results. In Section 5.3.2

we introduce the template that our protocols will follow, and then will describe how this template can be used to compute "interesting" functions. In Sections 5.3.3-5.3.6 we introduce the protocols for securely computing various distance functions.

5.3.1 On Using General SFE Results

Computational geometry problems are difficult in a secure framework because floating point arithmetic is very expensive in this framework, and so operations such as square root and division should be avoided. However, their inverses (multiplication and squaring) are much easier.

However, if a general Boolean circuit is built for computing distance functions then multiplication is required. The simplest circuits for k-bit multiplication require $O(k^2)$ gates. There are asymptotic improvements to these circuits, but they come at the cost of large constant factors; the asymptotically best of them (and the worst in terms of having impractically large constant factors) is a circuit of size $O(k \log k \log\log k)$ derived from the textbook Schoenhage-Strassen integer multiplication algorithm (which is itself of mainly theoretical interest, and not used in practice).

Now there are techniques based upon arithmetic circuits, which conceivably make multiplications a single operation (using homomorphic encryption). However, in such cases comparisons are not possible (and these are required by many of our protocols) unless the modulus is 2, which makes multiplication as expensive as the Boolean circuit case.

Thus the goal of our protocols will be to use arithmetic circuit based constructions for the multiplications and additions and then use Boolean circuit based constructions for the comparison. The difficulty of this approach is the conversion between the two forms, but in Sections 3.7.1 and 3.7.2 we introduced protocols that provide the necessary building blocks to perform this conversion efficiently.

5.3.2 Protocol Design

In the following Sections we give protocol for computing the distance between two objects (such as points, line segments, etc.). Given such protocols it is possible to construct other protocols based upon distance (using simple circuits to process the result). We now outline some examples of this:

1. Given a set of objects S and a specific object x, what is the nearest object in S to x?
2. Given two objects, are these two objects "close" to one another? This is useful for determining if a route is close to an object of interest or to determine if two routes are dangerously close.
3. Given a set of objects, are there any two objects that are close?

Thus all we need to develop are protocols to compute the distance between two objects. A problem with this is that many distance formulas require square roots and division. Square roots are handled by computing the square of the distance (notice that order is still preserved). To handle the division, we will store the distance as two integer values (the numerator and the denominator); it is still possible to compare these values (as both with be positive) by cross multiplying and then by comparing.

The communication complexity and computational complexity of these protocols are not that meaningful because all of the protocols consist of small constant number of scalar product protocols and scrambled circuit evaluation of simple circuits. However the round complexity is interesting, because many steps can be done in parallel. Thus when describing the complexity of our protocols we will consider only the round complexity, but we will describe the round complexity in exact terms rather than asymptotic terms (as $O(1)$ is not that meaningful in this context).

5.3.3 Distance between Two Points

The distance between P_1 and P_2 is $\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$, and thus we compute $(x_1 - x_2)^2 + (y_1 - y_2)^2$. This protocol is also presented in [105], but for completeness we give the protocol in Figure 5.1.

Input: Alice has point $P_1(x_1, y_1)$. Bob has a point $P_2(x_2, y_2)$.
Output: Computes (in a split manner) the distance between P_1 and P_2.
Steps:

1. Alice creates a vector $\vec{a} =< {x_1}^2, -2x_1, 1, {y_1}^2, -2y_1, 1 >$. Likewise, Bob creates a vector $\vec{b} =< 1, x_2, {x_2}^2, 1, y_2, {y_2}^2 >$.
2. Alice and Bob engage in protocols to compute $d = SCALARPRODUCT(\vec{a}, \vec{b})$ in a modular additively split manner.

Figure 5.1. Secure Protocol for Computing the Distance between Two Points

The above protocol requires a single round of interaction between Alice and Bob.

5.3.4 Distance between Point and Line Segment

In this Section we introduce protocols for computing the distance between a point and a line segment. The distance between a point and a line segment is either the distance between that point and an endpoint of the line segment or it is the distance between that point and an interior point of the segment. If it is the latter case, then this point must exist between the lines that are perpendicular to the segment and that are through the endpoints of the segment. To clarify this, Figure 5.2 contains all points that are within T units of a specific segment $\overline{AB}$.

Before we introduce a protocol for computing the distance between a point and a line segment, we need three building block protocols: i) one to compute the distance between two points (Figure 5.1), ii) one that computes the distance between a point and a line (Figure 5.3), and iii) one that determines if a point is between two lines (Figure 5.4).

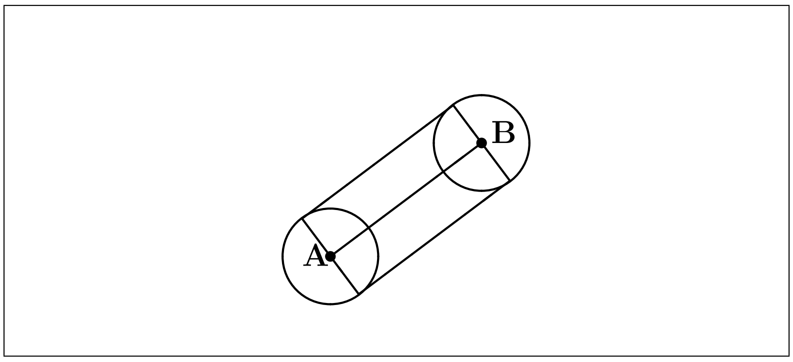

Figure 5.2. Area of all points within distance T from segment $\overline{AB}$

Distance between a Point and Line

The distance between $P(x, y)$ and $L(A, B, C)$ is $|\frac{Ax+By+C}{\sqrt{A^2+B^2}}|$. We compute the square of this distance, which is $\frac{(Ax+By+C)^2}{A^2+B^2}$. We give a protocol for computing this distance in Figure 5.3.

Input: Alice has point $P(x, y)$. Bob has line $L(A, B, C)$.
Output: Computes the distance between P and L in a split fashion.
Steps:

1. Alice creates vectors $\vec{a_1} =< x^2, 2xy, 2x, y^2, 2y, 1 >$ and $\vec{a_2} =< 1 >$. Likewise, Bob creates vectors $\vec{b_1} =< A^2, AB, AC, B^2, BC, C^2 >$ and $\vec{b_2} =< A^2 + B^2 >$.
2. Alice and Bob engage in protocols to compute the values $SCALARPRODUCT(\vec{a_1}, \vec{b_1})$ and $SCALARPRODUCT(\vec{a_2}, \vec{b_2})$, and they output these values as the distance.

Figure 5.3. Secure Protocol for Computing the Distance between a Point and a Line

Note that the above protocol computes the distance with a single round of interaction between Alice and Bob. Furthermore, the correctness of the protocol follows directly from its construction, and the privacy is as strong as the scalar product protocols.

Point between Two Lines

A point $P(x, y)$ is between two parallel lines, $L(A, B, C_1)$ and $L(A, B, C_2)$ where $C_1 < C_2$, if and only if $C_1 \leq Ax + By \leq C_2$. Figure 5.4 gives a protocol that computes this value in a XOR-split fashion.

Input: Alice has point $P(x, y)$. Bob has two parallel lines $L_1(A, B, C_1)$ and $L_2(A, B, C_2)$ (we assume that L_2 is above L_1 (i.e., $C_2 > C_1$)).
Output: Computes if P lies between L_1 and L_2.
Steps:

1. Alice creates a vector $\vec{a} =< x, y >$. Likewise, Bob creates a vector $\vec{b} =< A, B >$
2. Alice and Bob engage in protocols to compute $d = SCALARPRODUCT(\vec{a}, \vec{b})$ in a modular additively split fashion.
3. Alice and Bob compute $(d \geq C_1) \wedge (d \leq C_2)$ using SCE and output the result in XOR-split fashion.

Figure 5.4. Secure Protocol for Determining if a Point is between Two Lines

Note that the above protocol computes the result with a single scalar product protocol and a single circuit evaluation. If the protocol does not use base reduction, then it is possible to engage in this protocol with 2 rounds of interaction, and if base reduction is used then 3 rounds are required. Furthermore, the correctness of the protocol follows directly from its construction, and the privacy is as strong as the scalar product protocols and SCE.

Distance between a Point and a Segment

We are now ready to give the main protocol of this Section. As discussed earlier all that needs to be computed is the distance between the point and the endpoints of the segment and the distance between the point and the line through the segment. Furthermore, the latter distance is used only when the point in question is between the parallel lines that are through the endpoints of the segment and are perpendicular to the segment. Figure 5.5 describes this protocol.

Input: Alice has point $P(x_p, y_p)$. Bob has a line segment $S((x_1, y_1), (x_2, y_2))$.
Output: Computes the distance between P and S.
Steps:

1. Alice and Bob engage in secure protocol for computing the distance between (x_p, y_p) and (x_1, y_1) and store the result d_1 in a split fashion.
2. Alice and Bob engage in secure protocol for computing the distance between (x_p, y_p) and (x_2, y_2) and store the result d_2 in a split fashion.
3. Bob computes the line through his segment, which is $L(A, B, C)$ where $A = y_2 - y_1$, $B = x_1 - x_2$, and $C = x_1(y_1 - y_2) + y_1(x_2 - x_1)$. Alice and Bob engage in protocol to determine the distance between P and L and store the result d_3 in a split fashion.
4. Bob computes the perpendicular lines through the segment endpoints, denoted by $L_1 = (x_2 - x_1)x + (y_2 - y_1)y + (x_1 - x_2)x_1 + (y_1 - y_2)y_1$ and $L_2 = (x_2 - x_1)x + (y_2 - y_1)y + (x_1 - x_2)x_2 + (y_1 - y_2)y_2$. Alice and Bob engage in secure protocol to determine if P is between L_1 and L_2 and store the result γ in a XOR-split fashion.
5. Alice and Bob use SCE to compute the following distance in a split manner: if γ then output d_3 otherwise output $\min\{d_1, d_2\}$.

Figure 5.5. Secure Protocol for Computing the Distance between a Point and a Segment

Since many of the steps in the above protocol can be done in parallel, the round analysis of the protocol is non-trivial. However, steps 1 to 4 can be done in parallel with 2 rounds or 3 rounds if base reductions is used. Furthermore, the last step requires only a single round, and thus entire protocol requires 3 or 4 rounds. The security and correctness follow directly from the protocol.

5.3.5 Distance between Two Parametric Equations

Suppose Alice has a route which consists of a set of line segments and Bob has a route which consists of a set of line segments. Furthermore, suppose that there is an object that is traveling on each of Alice and Bob's routes whose position can be described by a point with constant velocity (with parametric equations). Alice and Bob would like to know the minimum distance between their objects.

General Idea

For this protocol we assume that Alice and Bob's time intervals are not secret (i.e., only the location, direction, and velocity during the intervals are private, but it is okay to reveal the time when the direction or velocity changes), and thus Alice and Bob can easily compute all time intervals where there is a change (in direction or velocity) by one of the objects. In this Section, we give a protocol to determine the minimum distance between Alice and Bob's objects (during the time interval). Suppose Alice's point can be described by $x_1(t) = v_{x,1}t + x_{0,1}$ and $y_1(t) = v_{y,1}t + y_{0,1}$, and likewise Bob's position can be described as $x_2(t) = v_{x,2}t + x_{0,2}$ and $y_2(t) = v_{y,2}t + y_{0,2}$. The distance between the points at any time t can be described by $\sqrt{(x_1(t) - x_2(t))^2 + (y_1(t) - y_1(t))^2}$. Thus our goal is to minimize a quadratic polynomial $At^2 + Bt + C$ inside of an interval $[s, e]$.

Quadratic polynomial minimization

The polynomial $f(t) = At^2 + Bt + C$ will have a minimum iff $A > 0$. If $A < 0$ it will be a downward parabola (i.e., it will have a maximum) and if $A = 0$ it will be a line (i.e., it will not have a maximum or minimum). If f has a minimum, then the value t that minimizes $f(t)$ can be easily found to be $t_{min} = \frac{-B}{2A}$. Assuming $A > 0$, then the minimum is in the interval $[s, e]$ iff $2As \leq -B \leq 2Ae$. Finally, the minimum value of $f(t)$ (assuming that there is such a minimum) is $f(t_{min}) = \frac{B^2}{4A} - \frac{B^2}{2A} + C$. We present this protocol in Figure 5.6.

As many of these steps can be done in parallel, the round complexity is nontrivial. Steps 1-3 can be done in parallel, and thus will require only 2 rounds without base reduction and 3 rounds with base reduction. The last round requires only 1 round of interaction, and thus the protocol requires only 3 or 4 rounds.

Input: Alice and Bob have a polynomial $f(t) = At^2 + Bt + C$ where $t \in [s, e]$ and where the values A, B, C, s, and e are additively split between them.
Output: If the minimum of $f(t)$ is in [s,e] output this minimum value otherwise output ∞.
Steps:

1. *Determine if a minimum exists to $f(t)$.* Alice and Bob engage in a protocol $\gamma_1 = (A > 0)$ in an XOR-split fashion.
2. *Assuming that there is a minimum determine if the minimum exists in $[s, e]$.* Alice and Bob engage in a protocol to determine: $\gamma_2 = (2As \leq -B) \wedge (-B \leq 2Ae)$ in an XOR-split fashion. Note that the values being compared can be computed with a secure scalar product.
3. *Compute the distance* Alice and Bob engage in a protocol to compute $\frac{-B^2+4AC}{4A}$ and call the result d in a split fashion.
4. *Output the final result:* Alice and Bob engage in a circuit that will output d if $\gamma_1 \wedge \gamma_2$ and will output ∞ otherwise.

Figure 5.6. Secure Protocol for Determining the Minimum of a Polynomial

Two Parameterized Line Segments

Suppose that the routes are paramaterized by time t and that they have constant velocity. The distance between the two points is $\sqrt{(x_A - x_B)^2 + (y_A - y_B)^2}$. All of these terms are linear in t (where t is time). This means that there is a polynomial $f(t) = At^2 + Bt + C$ such that the distance is $\sqrt{f(t)}$. Either $f(t)$ will have a global minimum in the interval $[s, e]$ or the minimum value in the interval is $\min\{f(s), f(e)\}$. Figure 5.7 gives a protocol for computing this distance.

The above protocol requires 5 or 6 rounds depending on whether or not base reduction is used. One round to compute the first and second steps, 3-4 rounds to do step 3, and another round to do step 4.

5.3.6 Distance between Two Line Segments

Suppose Alice has a route which consists of several line segments each associated with a distance and Bob has a route which consists of a series of segments. Further-

Input: Alice has segment $x_1(t) = v_{x,1}t + x_{0,1}$, $y_1(t) = v_{y,1}t + y_{0,1}$. Bob has a line segment $x_2(t) = v_{2,B}t + x_{0,2}$ and $y_2(t) = v_{y,2}t + y_{0,2}$. The time interval is known to Alice and Bob and is represented by s to e.
Output: Computes the minimum distance between the two segments during the time interval.
Steps:

1. Alice and Bob engage in scalar product protocols to determine the distance function $f(t) = At^2 + Bt + C$. The values A, B, and C are additively split between Alice and Bob. To avoid cluttering this exposition, we omit the exact details of these scalar product computations.
2. Alice and Bob determine the minimum distance at either endpoint of the interval:
   (a) Alice and Bob compute $d_s = As^2 + Bs + C$ and $d_e = Ae^2 + Be + C$ in an additively split fashion. Note that these values can easily be computed without communication.
3. Alice and Bob use Figure 5.6 with parameters A, B, C, s, and e to determine the minimum point inside the intervals. Alice and Bob store the result as d in a split manner.
4. Alice and Bob engage in a circuit to compute $\min\{d_s, d_e, d\}$.

Figure 5.7. Secure Protocol for Computing the Distance between Two Parameterized Line Segments

more, Alice and Bob want to know the distance between Alice and Bob's route. In this case, we need to be able to compute the distance between two segments.

General Idea

In two dimensions, either the segments will intersect or the closest point to the segment will involve one of the endpoints. Thus Figure 5.5 can be used 4 times (once with each endpoint and the other segment) to handle the second case. All that is needed is a protocol for determining if two segments intersect, for which a protocol is given in [105].

Line and Line Segment Intersection

L and S will intersect iff there is the endpoints of S lie on different sides of L or one or more of the endpoints is on L. The side of a point $P(x, y)$ on L is the sign of $Ax+By+C$. Thus, if the value of this is negative for one endpoint and is positive for the other, then the segment intersects the line. This protocol is described in Figure 5.8, and is also given in [105].

Input: Alice has line $L(A, B, C)$ and Bob has a line segment $S((x_1, y_1), (x_2, y_2))$.
Output: Computes if L and S intersect.
Steps:

1. Alice creates a vector $\vec{a} =< A, B, C >$. Likewise, Bob creates vectors $\vec{b_1} =< x_1, y_1, 1 >$ and $\vec{b_2} =< x_2, y_2, 1 >$.
2. Alice and Bob engage in protocols to compute $d_1 = SCALARPRODUCT(\vec{a}, \vec{b_1})$ in a modular additively split fashion.
3. Alice and Bob engage in a protocol to compute $d_2 = SCALARPRODUCT(\vec{a}, \vec{b_2})$ in a modular additively split fashion.
4. In parallel Alice and Bob engage in secure protocols to compute in an XOR-split fashion $\gamma_1, \gamma_2, \gamma_3, \gamma_4$ with respective values $(d_1 \leq 0), (d_2 \leq 0), (d_1 \geq 0), (d_2 \geq 0)$
5. Alice and Bob output $(\gamma_1 \wedge \gamma_4) \vee (\gamma_2 \wedge \gamma_3)$ in a XOR-split fashion.

Figure 5.8. Secure Protocol for Determining if a Line and a Line Segment Intersect

This protocol requires 2 or 3 rounds depending on whether or not base reduction is used.

Distance between Two Line Segments

All that needs to be done is to compute the distance between each endpoint and the other segment and to determine if the segments intersect. Figure 5.9 introduces this protocol.

Step 1 requires 3 or 4 rounds depending on whether or not base reduction is used, but Steps 2 and 3 can be done at this time also. Finally step 4 requires only a single round of interaction. Thus this protocol requires 4 or 5 rounds of interaction.

Input: Alice has segment $S_1(P_1(x_1, y_1), P_2(x_2, y_2))$ and a threshold T. Bob has a line segment $S_2(P_3(x_3, y_3), P_4(x_4, y_4))$.
Output: Computes the distance between S_1 and S_2.
Steps:

1. In parallel Alice and Bob engage in secure protocols to compute in a split fashion d_1, d_2, d_3, d_4 where the values are the result of execution of Protocol 5.5 (distance between a point and a line segment) with respective inputs (P_1 and S_2), (P_2 and S_2), (P_3 and S_1), and (P_4 and S_1).
2. Bob computes the line through his segment, which is $L_2(A, B, C)$ where $A = y_4 - y_3$, $B = x_3 - x_4$, and $C = x_3(y_3 - y_4) + y_3(x_4 - x_3)$. Alice and Bob engage in Protocol 5.8 on S_1 and L_2 storing the results in γ_1 in an XOR-split fashion.
3. Alice computes the line through her segment, which is $L_1(A, B, C)$ where $A = y_2 - y_1$, $B = x_1 - x_2$, and $C = x_1(y_1 - y_2) + y_1(x_2 - x_1)$. Alice and Bob engage in Protocol 5.8 on S_2 and L_1 storing the results in γ_2 in an XOR-split fashion.
4. Alice and Bob engage in secure protocols to output a distance d in a split fashion, where d is 0 if $(\gamma_1 \wedge \gamma_2)$ and is the minimum of d_1, d_2, d_3, and d_4 otherwise.

Figure 5.9. Secure Protocol for Determining the Distance between Two Line Segments

5.4 Perturbation-Based Techniques

In this Section, we investigate nearest neighbour queries (i.e., the post-office problem), and introduce several perturbation based techniques for protecting privacy including: random perturbation of the client's position, changing the user's query into a grid location, anonymizing the client's identity, asking multiple queries, and hybrid techniques. In many of these techniques, the client can use the protection without the cooperation from the server (i.e., the client can use the server even if all that the server does is answer nearest neighbour queries for points). Furthermore, some of the techniques introduce inaccuracies into the answer, but this error is controllable, and thus the client can choose a privacy-accuracy tradeoff. While these techniques are less private than the secure protocols outlined earlier, these perturbation based techniques are more efficient.

5.4.1 Random Perturbation of the Client's Position

This Section presents a solution that is simple in the sense that it does not require any modification to the way the database does its query-processing. It suffers from drawbacks that will be pointed out below.

Let $\vec{P}$ denote the position vector of the client, given as a pair of geographic coordinates. The database contains *sites* of the kind specified in the query (e.g., "gas station", or "post-office"), and the answer to the query is the database site nearest to $\vec{P}$. The results of this Section hold for any distance metric, not only the Euclidean one.

A simple solution consists of the following steps:

1. The client selects a distance δ large enough that the client deems it acceptable if his location was known by the remote database with an error of δ. Note that δ is not known to the database, and may vary from one query to the next even for the same client (because the privacy/accuracy tradeoff for that client may change over time, or from one query to the next).

2. The client generates a random vector $\vec{Q}$ of length $||\vec{Q}|| = \delta$, and sends as query the "fake" position $\vec{P} + \vec{Q}$.

3. The database responds with a position vector $\vec{R}$ of the database site nearest to $\vec{P} + \vec{Q}$.

The privacy parameter δ changes from one query to the next, depending on how important privacy is to the client at that moment of time, relative to the importance of an accurate answer (if he is very low on gas and wants the nearest gas station then he may choose $\delta = 0$).

We now quantify how much "damage" is done, to the quality of the answer, by the perturbation distance of δ. Let $\vec{S}$ be the true answer, the one that would be returned by the database if δ had been zero. The *damage* is the difference between the two distances $\vec{P}$-to-$\vec{R}$ and $\vec{P}$-to-$\vec{S}$ where $\vec{R}$ is the answer returned by the database in Step 3. In other words, the damage is

$$||\vec{P} - \vec{R}|| - ||\vec{P} - \vec{S}||.$$

Worst-case analysis

We now consider the worst-case value of the damage to the query's answer, that results from perturbing the query's location by δ.

Theorem 5.4.1 *If, in a nearest-neighbour proximity query, the client position is randomized by adding to it a random vector of length δ, then the damage to the answer is no greater than 2δ:*

$$||\vec{P} - \vec{R}|| - ||\vec{P} - \vec{S}|| \leq 2\delta.$$

Furthermore, this bound of 2δ is tight, i.e., there is an example where it is achieved.

Proof. We begin with the proof of the upper bound of 2δ on the damage. The vectors $\vec{P}$, $\vec{Q}$, $\vec{R}$, $\vec{S}$ are as defined in the above query-processing algorithm. If $\vec{R} = \vec{S}$ then the damage is zero, hence smaller than 2δ and the proof is done. So assume henceforth that $\vec{R} \neq \vec{S}$.
First, observe that:

$$||\vec{P} - \vec{R}|| = ||\vec{P} + \vec{Q} - \vec{Q} - \vec{R}|| \leq ||\vec{Q}|| + ||\vec{P} + \vec{Q} - \vec{R}|| = \delta + ||\vec{P} + \vec{Q} - \vec{R}|| \quad (1)$$

where the triangle inequality was used.
Now, the fact that the answer returned is $\vec{R}$ rather than $\vec{S}$ implies the following

$$||\vec{P} + \vec{Q} - \vec{R}|| \leq ||\vec{P} + \vec{Q} - \vec{S}||$$

which, using the triangle inequality, gives

$$||\vec{P} + \vec{Q} - \vec{R}|| \leq ||\vec{Q}|| + ||\vec{P} - \vec{S}|| = \delta + ||\vec{P} - \vec{S}|| \quad (2)$$

Combining (1) and (2) gives

$$||\vec{P} - \vec{R}|| \leq \delta + \delta + ||\vec{P} - \vec{S}||$$

which gives

$$||\vec{P} - \vec{R}|| - ||\vec{P} - \vec{S}|| \leq 2\delta.$$

This completes the proof of the upper bound.

We now given an example that achieves the bound of 2δ just proved. It suffices to give a one-dimensional example (in which case the vectors are scalars). Choose $S = 0$, $Q = +\delta$, $R = 6\delta$, and $P = 2\delta + \epsilon$ where ϵ is arbitrarily small (much smaller than δ). The database gets queried for the site nearest to $P + Q = 3\delta + \epsilon$, so it returns the site $R = 6\delta$. The site nearest to P is $S = 0$. The damage in that case is

$$|P - R| - |P - S| = |2\delta + \epsilon - 6\delta| - (2\delta + \epsilon) = (4\delta - \epsilon) - 2\delta - \epsilon = 2\delta - 2\epsilon$$

which goes to 2δ as ϵ is made arbitrarily small. □

Average-case analysis

The case of worst-case damage is unlikely to occur, in fact it has measure zero in a probabilistic model of random (uniformly distributed) sites, queries, and "perturbation vectors" $\vec{Q}$ (with a fixed modulus δ for the perturbation, so only its direction is random). In such a probabilistic model, what is the expected value of the damage? This Section provides an answer, under one assumption we need to make so as to handle the case of the perturbation vector $\vec{Q}$ taking the query "outside the map", i.e., outside the region in which the n sites lie; to avoid this troublesome situation, and for the sake of making the analysis tractable, we henceforth assume that the map wraps around as on the surface of a sphere (hence no perturbation can result in crossing the map's boundary, as there is no boundary). If the map in reality does have a boundary, then our analysis still has some validity as an approximation because what it means then is that we are only concerning ourselves with points that are "well within" the bounding box containing the n sites, that we exclude from the analysis "boundary effects" caused by perturbations that cause boundary-crossing, or due to sites close to the boundary of the bounding box. In such a case, the justification for excluding the boundary sites from the analysis is that they are a negligible fraction of the total number n of sites – in fact they are zero percent for an infinitely large n (the ratio of boundary to total has a $\sqrt{n}$ in its denominator and hence goes to zero as n goes to infinity).

Theorem 5.4.2 *The expected damage is no larger than δ, that is,*
$E(||\vec{P}-\vec{R}|| - ||\vec{P}-\vec{S}||) \leq \delta$.

Proof. First, observe that the "nearest-neighbour" distance statistics are location-independent. This implies the following

$$E(||\vec{P}-\vec{S}||) = E(||\vec{P}+\vec{Q}-\vec{R}||) \tag{3}$$

because $\vec{S}$ is to $\vec{P}$ what $\vec{R}$ is to $\vec{P}+\vec{Q}$ (namely, the site nearest to it). Now observe that

$$||\vec{P}-\vec{R}|| = ||\vec{P}+\vec{Q}-\vec{Q}-\vec{R}|| \leq ||\vec{Q}|| + ||\vec{P}+\vec{Q}-\vec{R}|| \tag{4}$$

where the triangle inequality was used. Taking expectations in Equation (4) and then using Equation (3) gives

$$E(||\vec{P}-\vec{R}||) \leq \delta + E(||\vec{P}-\vec{S}||)$$

and we therefore have:

$$\text{Expected Damage} = E(||\vec{P}-\vec{R}|| - ||\vec{P}-\vec{S}||) \leq \delta$$

which completes the proof. □

Discussion of Location Perturbation

The disadvantages of the above technique are: i) the server learns the proximity of the client and ii) there are inaccuracies in the answer (albeit inaccuracies that are controllable by the client). However, the client can use this approach without cooperation from the server and the scheme is very efficient.

5.4.2 Grid Method

In this Section, we present a variation on the scheme presented in the previous Section: Unlike the previous Section, this variant does not result in any loss of accuracy, but it potentially requires more communication. The idea behind this scheme is to "grid" the plane, covering it with tiles of dimensions $\lambda \times \lambda$; after this gridding of the plane, the client queries the database with the tile that contains the client's location. The database answers the query with all sites that are closest to at least one point in the query tile; that is, if v is any point of the query tile (not necessarily a site) and site w is the closest site to v, then w is a part of the answer that the database will return to the client (note that w could be inside the query tile, or outside of it, and that a site inside the query tile is always chosen as a part of the answer). Upon receiving these sites the client determines which of them is closest to his actual location. The disadvantage of this scheme is that the client has potential to receive many sites in response to the query – the expected number received depends on λ but also on the average density ρ of sites per unit area (the two determine the expected number of sites per tile, which is $\lambda^2\rho$). Note that, if the n points are inside a $\Delta \times \Delta$ bounding box, then $\rho = n/\Delta^2$. It would be interesting to determine precisely the expected number of sites returned (with the correct constant factors), assuming a randomly selected query tile and (as usual) uniformly distributed sites. Since we already know that points inside the query tile are always included in the answer, their expected number $\lambda^2\rho$ $(= n\lambda^2/\Delta^2)$ is a lower bound on the expected number of sites returned.

A refinement of the above scheme is to have the database treat the answers that would be returned by the above scheme merely as "candidates" for the one site that is returned as answer: The site that has the largest number of "votes" from within the tile. In other words, if v and w are as above, then the winning candidate w is the one with the largest number of v's in the tile that "choose it" as the nearest site to them. This variant, which we call *one-answer variant* of the grid method, does not have the increased communication because a single site is returned as answer, but it does have an accuracy tradeoff that is quantified below.

Worst-Case Analysis of One-Answer Variant

Theorem 5.4.3 *In the grid method, the worst-case damage to a query's answer is no greater than twice the tile diameter. Furthermore, this bound is tight, i.e., there is an example where it is achieved.*

Proof. Same as for Theorem 1, except that $||\vec{Q}||$ is no longer δ but rather the random distance between the actual client location $\vec{P}$, and the location $\vec{P'}$ within the tile that is "responsible" for the fact that it is the site $\vec{R}$ that was returned as the answer; that is, $\vec{Q} = \vec{P'} - \vec{P}$. Hence $||\vec{Q}||$ is, in the worst case, the diameter of a $\lambda \times \lambda$ tile. □

Corollary 1 *In the grid method and using the Euclidean distance metric, the worst-case damage to a query's answer is no greater than* $2\sqrt{2}\lambda$. *Using the Manhattan distance* L_1 *the worst-case damage is no more than* 4λ. *Using the* L_∞ *distance it is no more than* 2λ.

Alternative Proof:

Theorem 5.4.4 *In the grid method, the worst-case damage to a query's answer is no greater than the tile diameter. Furthermore, this bound is tight, i.e., there is an example where it is achieved.*

Proof. Clearly, the answer that is returned will be the location that is closest to the centroid of the tile. If the answer that is returned is inside the tile then this bound is obviously the maximum distance between two points in the tile is no larger than the diameter of the tile (from now on the diameter for the tile is referred to as d). We now consider points outside the tile.

In what follows we use $\vec{P}$ to denote the client's location vector, $\vec{Q}$ to denote the tile's centroid vector, $\vec{R}$ to denote the database's answer vector, and $\vec{S}$ to denote the true answer to the client's query.

Now, all that we must show is that $||\vec{P}-\vec{R}|| - ||\vec{P}-\vec{S}|| \leq d$. From the triangle inequality,

$$||\vec{P}-\vec{R}|| = ||\vec{P}+\vec{Q}-\vec{Q}-\vec{R}|| \leq ||\vec{P}-\vec{Q}|| + ||\vec{Q}-\vec{R}|| \leq \tfrac{d}{2} + ||\vec{Q}-\vec{R}|| \qquad (5)$$

and

$$||\vec{P}-\vec{S}|| = ||\vec{P}+\vec{Q}-\vec{Q}-\vec{S}|| \geq ||\vec{Q}-\vec{S}|| - ||\vec{P}-\vec{Q}|| \geq ||\vec{Q}-\vec{S}|| - \tfrac{d}{2} \qquad (6)$$

Thus combining equations (5) and (6) gives

$$||\vec{P}-\vec{R}|| - ||\vec{P}-\vec{S}|| \leq$$
$$(\tfrac{d}{2} + ||\vec{Q}-\vec{R}||) - (||\vec{Q}-\vec{S}|| - \tfrac{d}{2}) =$$
$$d + ||\vec{Q}-\vec{R}|| - ||\vec{Q}-\vec{S}|| \leq d$$

We now give an example that achieves the bound of the diameter of the tile that was just proved. It suffices to give a one-dimensional example (in which case the vectors are scalars). Choose $S = \lambda - \epsilon$, $P = 2\lambda$, $R = 4\lambda$, and $Q = 2.5\lambda$ where ϵ is arbitrarily small. The database receives a query for the tile $[2\lambda, 3\lambda]$ so it returns the site $R = 4\lambda$. The site nearest to P is $S = \lambda - \epsilon$. The damage in that case is

$$|P-R| - |P-S| = |2\lambda - (4\lambda)| - |(2\lambda - (\lambda - \epsilon)| = (2\lambda) - (\lambda + \epsilon) = \lambda - \epsilon$$

which goes to λ (which is the diameter of the tile in one dimension) as ϵ is made arbitrarily small. □

Average-Case Analysis of One-Answer Variant

The average-case damage, on the other hand, is not the same as in Theorem 2 with δ replaced by the tile diameter, because that δ was fixed, whereas in this case the "implied perturbation" is random. The following theorem assumes the Euclidean distance.

Theorem 5.4.5 *In the grid method and using the Euclidean distance metric, the expected damage to a query's answer is no larger than $c\lambda$ where*

$$c = (2 + \sqrt{2} + 5\ln(1+\sqrt{2}))/15 = 0.521405\cdots$$

Proof. In what follows we use $\vec{P}$ to denote the client's location vector, $\vec{P'}$ to denote the location of that point in the tile that is responsible for causing the database to return $\vec{R}$ as answer, and $\vec{S}$ to be the true answer to the query. As in the proof of Theorem 2 (and under the same wraparound assumption as in that proof), we use the fact that the "nearest-neighbour" distance statistics are location-independent, which implies the following

$$E(||\vec{P} - \vec{S}||) = E(||\vec{P'} - \vec{R}||) \tag{5}$$

because $\vec{S}$ is to $\vec{P}$ what $\vec{R}$ is to $\vec{P'}$ (namely, the site nearest to it). Now observe that

$$||\vec{P} - \vec{R}|| = ||\vec{P} + \vec{P'} - \vec{P'} - \vec{R}|| \leq$$
$$||\vec{P} - \vec{P'}|| + ||\vec{P'} - \vec{R}|| \tag{6}$$

where the triangle inequality was used. Taking expectations in Equation (6) and then using Equation (5) gives

$$E(||\vec{P} - \vec{R}||) \leq E(||\vec{P} - \vec{S}|| + E(||\vec{P} - \vec{P'}||)$$

and therefore

$$\text{Expected Damage} = E(||\vec{P} - \vec{R}|| - ||\vec{P} - \vec{S}||) \leq E(||\vec{P} - \vec{P'}||).$$

Note that $E(||\vec{P} - \vec{P'}||)$ is the expected distance between two random points in a $\lambda \times \lambda$ square. It is well known [106] that the expected distance between two random points in a unit square is equal to the c in this theorem's statement, which implies

$$E(||\vec{P} - \vec{P'}||) = c\lambda$$

and the proof is complete. □

Discussion of the Grid Method

Comparing the grid method to the random perturbation method of the previous Section, we note that the λ for the grid method has to be known to the database (otherwise it cannot process the query), whereas the δ of the perturbation method was not known to the database.

The grid method is also more rigid for the following practical reasons. If the tiling is rigid and λ fixed, then the database can pre-compute, off-line, the site-proximity set of each tile, making it possible to subsequently perform on-line query-processing very fast (because processing a query tile is then a simple lookup operation). It is, however, expensive (in terms of query-processing computations at the database's end) to allow the client to dynamically change the value of λ from one query to the next – this would mean more computational overhead that cannot be pre-computed by the database. (As mentioned earlier, the client may wish to adjust the λ parameter from one query to another, as his privacy/accuracy tradeoff valuation changes.) One compromise solution would be to have a fixed menu of k available λ values that the client could choose from, which would combine giving more flexibility to the client while the database retains the ability to do off-line pre-processing of each tiling's site-proximity information; of course the database now has k separate tilings to maintain.

5.4.3 Anonymization

Another approach consists of accurately revealing to the database the client's location but hiding from the database the identity of the client. This can be done by interposing an anonymizing intermediary – a mix [107] – between the clients and the database, so the database knows the exact client location for each query but does not know who asked the query (thus there is no damage to the quality of the answer to the query). The intermediary knows the client is asking *some* query from the database but does not learn anything about the nature of the query or its parameters because the client-to-database traffic is encrypted [107] in both directions after the establishment of a session key through the usual cryptographic techniques [7]. Achieving the necessary "hiding in a crowd" effect, that is needed for foiling traffic analysis, requires that queries from many clients are handled by the same intermediary. Making such an intermediary act like a mix [107] provides resistance to traffic analysis: An eavesdropper could not make the connection between the traffic incoming from the many clients and the queries outgoing to the (possibly many) database(s).

The main problem with this kind of approach is its vulnerability to misbehavior by an intermediary: Even though the use of encryption easily hides from the intermediary the query coming from each client (only the database can decrypt it), all is lost if the intermediary were to collude with the database (the database could then associate the client's query and location with the client's identity). A cascade of mixes increases the security (collusion by all intermediaries would be needed for the system to fail), but also the cost and complexity.

Another drawback is that implementing a mix is difficult. To achieve a sort of "hiding in a crowd" without a mix, a simple solution would be for the client to send several locations to the database. All that the database learns is that the client is at one of several locations. The advantages of this scheme are its simplicity in that the database does not need to do anything other than answer queries. However, the disadvantages are that the level of privacy-protection is minimal and the communication complexity is larger than previous solutions.

The above approach could be modified to use the techniques in Section 5.4.1 (i.e., the client sends several perturbed points to the database). In a naive hybrid approach, the client would send multiple queries that are perturbations of his actual location, and the client would choose the resulting answer that was closest to his true location. This technique is flawed because the server can take the centroid of the points to obtain an accurate estimate of the client's location. A better approach is to perturb the client's location and then choose a set of perturbed points from this location. This provides a higher level of privacy and accuracy than the previous techniques, but comes at a cost in performance.

5.5 Summary

In this Chapter we have introduced privacy-preserving computational geometry protocols with applications to route planning. These applications include: planning of routes that do (or do not) get close to certain objects, planning routes of objects with constant velocities that do not get within a certain distance of each other, and planning routes of objects that do not get within certain distances of each other without knowing the exact velocity of the objects beforehand. We have also explored various perturbation based techniques that the client can use without a cooperating server. These techniques are more efficient than standard secure protocols, but this comes at the expense of accuracy, privacy, or communication. However, we believe that there are situations where these latter techniques are useful.

6 CONTRACT NEGOTIATION

6.1 Introduction

Suppose Alice and Bob are two entities who are negotiating a joint contract, which consists of a sequence of clauses (i.e., terms and conditions). Alice and Bob are negotiating the specific value for each clause. Example clauses include:

1. How will Alice and Bob distribute the revenue received for jointly performing a task?
2. Given a set of tasks, where Alice and Bob each have a set of tasks that they are willing and able to perform, who performs which tasks?
3. Given a set of locations to perform certain tasks, in which locations does Alice (Bob) perform her (his) tasks?

Alice and Bob each have private constraints on the acceptability of each clause (i.e., rules for when a specific term is acceptable). A specific clause is an agreement between Alice and Bob if it satisfies both of their constraints. In a non-private setting, Alice and Bob can simply reveal their constraints to one another. However, this has two significant drawbacks: i) if there are multiple possible agreements how do Alice and Bob choose a specific agreement (some are more desirable to Alice, others are more desirable to Bob), and ii) the revelation of one's constraints and preferences is unacceptable in many cases (e.g., one's counterpart in the negotiation can infer information about one's strategies or business processes). This second problem is exacerbated when Alice and Bob are competitors in one business sector but cooperate in another sector. We propose a framework and protocols that facilitate contract negotiation without revealing private constraints. There are two components to such a negotiation: i) the ability to determine if there is a contract that satisfies both parties' constraints (without revealing anything other than "yes/no") and ii) if there is a contract that satisfies both parties' constraints, then a protocol for determining a contract that is *valid* (acceptable to both parties), *fair* (when many valid outcomes are possible one of them is selected randomly with a uniform distribution, without either party being able to control the outcome), and *efficient* (no clause is replaceable by another that is better for both parties).

We introduce protocols for both of these tasks in the semi-honest model (i.e., the parties will follow the protocol steps but may try to learn additional information). The results of the Chapter are summarized as follows:

- The definition of a framework for privacy preserving contract negotiation. This framework allows multiple independent clauses, but can be extended to support dependencies between clauses.
- Protocols for determining if there is a valid contract according to both parties' constraints.
- Protocols for determining a fair, valid, and efficient contract when there is such a contract. The most difficult of these requirements is fairness, and we believe that the ability to choose one of several values without either party having control of the value will have applications in other domains.

The rest of the Chapter is organized as follows. In Section 6.2, an overview of related work is given. Section 6.3 outlines the framework for secure contract negotiation. Section 6.4 describes protocols for computing the satisfiability of a clause as well as for computing a fair term for a clause. In Section 6.5, we discuss extensions to our protocols that allow Alice and Bob to make preferences. Section 6.6 introduces several other extensions to our framework. Finally, Section 6.7 summarizes our results.

6.2 Related Work

The authors are not aware of any directly related work in this area. Much of the previous work in automated contract negotiation [108,109] focuses on creating logics to express contract constraints so that agreements can be computed. Our work is not to be confused with simultaneous contract signing [7], which solves the different problem of achieving simultaneity in the signing of a pre-existing already agreed-upon contract. The closest work is in [110], which deals with user preference searching in e-commerce. The problem addressed here is that a vendor may take advantage of a customer if that vendor learns the customer's constraints on a purchase (type of item, spending range, etc.). To prevent this, [110] suggests using a gradual information release.

6.3 Secure Contract Framework

In this Section we introduce a framework for secure contract negotiation. We begin with several definitions:

- A *clause* is a public set $S = \{s_0, \dots s_{N-1}\}$ of possible values. We refer to each of these values as *terms*. We assume that Alice and Bob can agree on S at the start of the negotiation. Furthermore, there is a defined ordering of the terms, so that s_i is the ith term in the set.
- For each clause S, Alice (Bob) has a set of *constraints* on the acceptability of each of that clause's terms. These constraints are represented by sets A (respectively, B), where $A \subseteq S$ ($B \subseteq S$) and A (B) is the set of all terms for clause S that are acceptable to Alice (Bob).
- A term $x \in S$ is *acceptable* iff $x \in (A \cap B)$.
- A clause is *satisfiable* iff $A \cap B \neq \emptyset$, i.e., there is a term for the clause that is acceptable to both Alice and Bob.

- A *negotiation* is a sequence of clauses $S_0, \ldots, S_{k-1}$. In this paper, we assume that these clauses are independent (i.e., that the acceptability set of one clause does not depend on the outcome of another clause). We briefly discuss how to extend our protocols for dependent clauses in Section 6.6.3. A negotiation is *satisfiable* iff each clause is satisfiable.

- A *contract* for a negotiation is a sequence of terms $x_0, \ldots, x_{k-1}$ (where $x_i \in S_i$). A contract is *valid* if each term is acceptable to both parties. A valid contract is *efficient* if no term in it is replaceable by another term that is better for both Alice and Bob (according to their respective private valuation functions).

Example 7 Suppose Alice and Bob are entities that jointly manufacture some type of device, and furthermore they must negotiate where to manufacture the devices. The clause could take the form $S = \{$London, New York, Chicago, Tokyo, Paris, Ottawa$\}$. Now suppose Alice's constraints are $A = \{$London, New York, Paris, Ottawa$\}$ and Bob's constraints are $B = \{$London, Tokyo, Paris, Ottawa$\}$. The set of acceptable terms are those in $A \cap B = \{$London, Paris, Ottawa$\}$.

We now outline the framework for our protocols. Protocols for computing the satisfiability of a clause and an acceptable term for the clause are given in the next Section. However, this needs to be extended to the contract level, because the individual results for each clause cannot be revealed when the negotiation is not satisfiable. Given a protocol that evaluates whether or not a clause is satisfiable in a split manner, then it is possible to determine if the contract is satisfiable. A contract is satisfiable if and only if all clauses are satisfiable, which can be computed easily by computing the AND of many split Boolean values using Scrambled Circuit Evaluation. A key observation is that if a negotiation is satisfiable, then to find a valid and fair contract one can find the individual valid and fair clause agreements independently. Thus given secure protocols for determining whether a clause is satisfiable and a protocol for determining an acceptable fair term for a satisfiable clause, it is possible to compute these same values at the negotiation level.

6.4 Secure Contract Term Protocols

In this Section, we introduce private protocols for computing: i) the satisfiability of a clause(yes/no) and ii) a fair agreement for satisfiable clauses (recall that fair means that the term is chosen uniformly from the set of all acceptable terms and that neither party has control over the outcome). We postpone discussion of protocols for computing efficient agreements until Section 6.5. We now define some notation for our protocols. We assume that Alice and Bob are negotiating a specific clause with N terms. We represent Alice's (Bob's) acceptability for ith term as a Boolean value a_i (b_i).

6.4.1 Determining Satisfiability

A clause is satisfiable iff $\bigvee_{i=0}^{N-1} a_i \wedge b_i$ is true. Clearly, this satisfiability predicate be computed with Scrambled Circuit Evaluation with $O(N)$ communication and $O(1)$ rounds. It is worth noting that Set Disjointness trivially reduces to this problem (in fact it really is the same problem), and thus there is an $\Omega(N)$ communication lower bound on this protocol [64].

6.4.2 Computing a Fair Acceptable Term

In this Section we introduce a protocol for computing a fair acceptable term for a clause that is known to be satisfiable. Figure 6.1 describes the protocol template for computing these values.

Input: Alice has a set of binary inputs $a_0, \ldots, a_{N-1}$ and likewise Bob has a set of inputs: $b_0, \ldots, b_{N-1}$. Furthermore it is known that $\exists i \in [0, N)$ such that $a_i \wedge b_i$ is true.
Output: An index j such that $a_j \wedge b_j$ is true, and if there are many such indices, then neither party should be able to control which index is chosen (by modifying their inputs).

Figure 6.1. Protocol Description

Figure 6.2 describes our protocol for computing a fair acceptable term. However, we also discuss three elements about the protocol's difficulty including: i) we show that semi-honest chosen OT reduces to the above mentioned problem, ii) we discuss a solution using circuit simulation for this problem, and iii) we show a false start for this problem that demonstrates the difficulty of a sub-linear communication protocol.

A Reduction from Semi-Honest OT

Suppose Bob is choosing 1 out of N items (item i) from Alice's list of binary values $v_0, \ldots, v_{N-1}$. Alice and Bob define a list of $2N$ values. Alice creates a list where item a_{2j+v_j} is true and a_{2j+1-v_j} is false for all $j \in [0, N)$. Bob creates a similar list, but sets only values b_{2i} and b_{2i+1} to true (and all other values are set to false). Alice and Bob engage in the above mentioned protocol. Clearly from the returned value, Bob can deduce the value of Alice's bit v_i.

Using Circuit Simulation

In order to make the term fair, the participants could each input a random permutation into the circuit that would compose the permutations and then permute the list with the composed permutation. The circuit would then choose the first value in the list that was an agreement. This would be fair because if at least one party chose a random permutation than the composed permutation would also be random (making the first acceptable item a fair choice). However, this would require at least $O(N \log N)$ inputs into the circuit (and thus this many 1-out-of-2 OTs) as this is the minimum number of bits to represent a permutation. Also, the circuit would have to perform the permutation, which would involve indirectly accessing a list of size N exactly N times. The direct approach of doing this would require $O(N^2)$ gates. Thus the standard circuit would require at least $O(N \log N)$ OTs (also this many modular exponentiations) and $O(N^2)$ communication. It may be possible to reduce the number of bits input into the circuit to $O(N)$ by using a pseudorandom permutation, however this would require the computation of a permutation, which would be a difficult circuit to construct. The protocol we describe requires only $O(N)$ modular exponentiations, $O(N)$ communication, and $O(1)$ rounds.

Input/Output: See Figure 6.1.

1. Alice does the following:
   (a) She chooses a semantically-secure homomorphic encryption function E_A (with modulus M_A) and publishes its public keys and public parameters.
   (b) For each item a_i in the list $a_0, \ldots, a_{N-1}$, she creates a value: $\alpha_i \leftarrow E_A(a_i)$. She sends these values to Bob.
2. Bob does the following:
   (a) He chooses a semantically-secure homomorphic encryption function E_B (with modulus M_B) and publishes the public keys and the public parameters.
   (b) For each i from 0 to $N-1$, Bob chooses/computes:
      i. A random value r_i chosen uniformly from $\{0,1\}$.
      ii. If $b_i = 0$, then he sets $\beta_i \leftarrow E_A(0)$, and otherwise he sets it to $\beta_i \leftarrow \alpha_i * E_A(0)$.
      iii. if $r_i = 0$, then $\gamma_i = \beta_i$, and otherwise $\gamma_i = ((\beta_i * E_A(M_A - 1))^{M_A - 1})$.
      iv. $\delta_i[0] \leftarrow E_B(r_i)$ and $\delta_i[1] \leftarrow E_B(1 - r_i)$

   Bob forms ordered triples $(\gamma_i, \delta_i[0], \delta_i[1])$ and randomly permutes all of the tuples (storing the permutation Π_B), and he sends the permuted list of ordered triples to Alice.
3. Alice permutes the triples using a random permutation Π' and then for each triple in the permuted list $(\gamma_i, \delta_i[0], \delta_i[1])$ (note that these i values are not the same ones that Bob sent, but are the new values in the permuted list) she computes/chooses:
   (a) $\zeta_i \leftarrow \delta_i[D_A(\gamma_i)] * E_B(0)$
   (b) $\eta_i \leftarrow \zeta_i * (\eta_{i-1})^2$ (if $i = 0$, then she sets it to ζ_0).
   (c) She chooses a random q_i uniformly from $\mathbb{Z}^*_{M_B}$.
   (d) $\theta_i \leftarrow (\eta_i * E_B(-1))^{q_i}$

   Alice permutes the θ values using another random permutation Π'' and she computes the permutation $\Pi_A = \Pi''\Pi'$. She sends the permuted θ values along with the permutation Π_A
4. Bob decrypts the values with D_B and finds the value that decrypts to 0; he then finds the original index of this value by inverting the permutation and he announces this index.

Figure 6.2. Protocol FIND-AGREEMENT

Some False Starts for Sub-Linear Communication

It would be desirable for a protocol to have sub-linear (in terms of the number of possible terms) communication. A possible strategy for this is to use a randomized approach. This solution works well if it is known that the sets have a "substantial" intersection, but all that is known is that there is at least one item in the intersection. Furthermore, the participants do not want to leak additional information about their own sets, including information about their mutual intersection. And thus any probabilistic strategy must behave as if there is only a single item in the intersection, and such a protocol would not have sub-linear communication. As a final note if the contract is to be fair and efficient then there is a lower bound on the communication complexity of $\Omega(N)$ (we prove this in Section 6.5).

Proof of Correctness

Before discussing the security of the above protocol, we show that the protocol is correct in that it computes an agreement. It is easy to verify that the permutations do not affect the result as they are reversed in the opposite order that they were used, and thus our correctness analysis ignores the permutations. We consider a specific term with respective acceptability for Alice and Bob as a_i and b_i (we use c_i to denote $a_i \wedge b_i$). We now trace the protocol describing each variable:

1. The value α_i is $E_A(a_i)$.
2. The value β_i is $E_A(c_i)$.
3. It is easy to verify that the value γ_i is $E_A(c_i \oplus r_i)$ (where $\oplus$ denotes exclusive-or).
4. The value $\delta_i[0]$ is $E_B(r_i)$ and the value $\delta_i[1]$ is $E_B(1 - r_i)$
5. Now, ζ_i is $\delta_i[0]$ when $c_i = r_i$ and is $\delta_i[1]$ otherwise. This implies that $\zeta_i = E_B(c_i)$.

6. Let $\hat{i}$ be the first index where $\zeta_{\hat{i}}$ is $E_B(1)$. For $i < \hat{i}$, the value η_i will be $E_B(0)$. Furthermore, the value $\eta_{\hat{i}}$ will be $E_B(1)$. However, for $i > \hat{i}$ the value η_i will be something other than $E_B(1)$, because $\eta_i = \zeta_i + {\eta_{i-1}}^2$.

7. If $\eta_i = E_B(x_i)$, the value θ_i will be $E_B(q_i(x_i - 1))$, this value will be $E_B(0)$ only when $x_i = 1$, which will only happen at $i = \hat{i}$. $\square$

Proof of Security (semi-honest model)

There are two parts to proving that this protocol is secure: i) that Alice (or Bob) does not learn additional information about indices that are not the output index, and ii) since we consider the permutations to be inputs into the protocol, we must show that that a party cannot choose its permutation to affect the outcome. Since Alice and Bob's roles are not symmetrical in the protocol, we must prove security for both cases.

Alice

We introduce an algorithm S_B (mnemonic for "Simulate Bob") that takes Alice's inputs and creates a transcript that is computationally indistinguishable from Alice's view in the real model. This algorithm is shown in Figure 6.3.

Input: Alice sees $a_0, \ldots, a_{N-1}, E_A, D_A, E_B$ and she sees the output index j.
Output: Alice must generate values indistinguishable from Bob's values in step 2; these values are triples of the form $(\gamma_i, \delta_i[0], \delta_i[1])$

1. Alice generates a sequence of values $\hat{b}_0, \ldots, \hat{b}_{N-1}$ where where $\hat{b}_i$ is chosen uniformly from $\{0, 1\}$ if $i \neq j$ and is 1 if $i = j$.
2. Alice generates a sequence of random bits $r_0, \ldots, r_{N-1}$ chosen uniformly from $\{0, 1\}$.
3. Alice creates tuples of the from $(E_A(r_i \oplus (a_i \wedge \hat{b}_i)), E_B(r_i), E_B(\neg r_i))$.
4. Alice permutes the items using a random permutation and then outputs these tuples.

Figure 6.3. Algorithm S_B

Lemma 1 *In the semi-honest model, S_B is computationally indistinguishable from Alice's view from running FIND-AGREEMENT.*

Proof: Since E_B is semantically-secure, the second and third elements of the tuple are indistinguishable from the real execution. To show that the first item is computationally indistinguishable, we must show two things: i) that the decrypted values are indistinguishable (since Alice knows D_A), and ii) that from Alice's previous information that she created in Step 1 she cannot distinguish the values.

To show (i), the decrypted values from S_B are chosen uniformly from $\{0, 1\}$. In the real execution the values are $E_A(c_i \oplus r_i)$, where r_i is chosen uniformly from $\{0, 1\}$. Thus, the sequences are indistinguishable.

Part (ii) follows from the statement that Bob performs at least one multiplication on each item or he generates the values himself. By the properties of semantically secure homomorphic encryption, this implies that these values are indistinguishable. $\square$

Lemma 2 *In the semi-honest model, Alice cannot control which term is chosen by selecting her permutation in the protocol FIND-AGREEMENT.*

Proof: The composition of two permutations, with at least one being random, is random. Thus, when Bob randomly permutes the tuples in Step 2, Alice cannot permute them in a way that benefits her, as she does not know Bob's permutation. Thus, when she computes the θ values the permutation is random and the term is chosen fairly.

$\square$

Bob

We introduce an algorithm S_A (mnemonic for "Simulate Alice") that takes Bob's inputs and creates a transcript that is computationally indistinguishable from Alice's view in the real model. This algorithm is shown in Figure 6.4.

Lemma 3 *In the semi-honest model, S_A is computationally indistinguishable from Bob's view from running FIND-AGREEMENT.*

Proof: Since E_A is semantically-secure, the values $E_A(\hat{a}_0), \ldots, E_A(\hat{a}_{N-1})$ are indistinguishable from the real execution and the permutation $\hat{\Pi}$ is also indistinguishable. To show that the the values $\hat{\theta}_0, \ldots, \hat{\theta}_{N-1}$ are computationally indistinguishable from the real execution, we must show two things: i) that the decrypted values are indistinguishable (since Bob knows D_B), and ii) that from his computations from Step 2, he cannot distinguish the values.

To show (i), the values in S_A are $N-1$ random values and a single 0 value where the 0 is placed randomly. And since in Step 3.d of the protocol, Alice multiplies the values by a random value and then permutes the items these values are indistinguishable.

To show (ii), all that needs to be shown is that Alice restores the semantic security property, which is clearly done in Step 3.a of the protocol, when she multiplies by $E_B(0)$. $\square$

Lemma 4 *In the semi-honest model, Bob cannot control which term is chosen by selecting his permutation in the protocol FIND-AGREEMENT.*

Proof: The composition of two permutations, with at least one being random, is random. Thus when Alice permutes the list with Π' the values are randomly permuted, and when the first agreement is chosen from this list, it is fairly chosen. $\square$

6.5 Expressing Preferences

It is of course unrealistic to assume that Alice and Bob have sets of acceptable states that are all equally desirable. There can be many terms for a clause that are a win-win situation for Alice and Bob (i.e., both prefer a specific term), however the random selection provided by FIND-AGREEMENT does not allow the choice of

Input: Bob sees $b_0, \ldots, b_{N-1}, E_B, D_B, E_A$ and he sees the output index j.
Output: Bob must generate values indistinguishable from Alice's values in steps 1 and 3; these values include: $E_A(a_0), \ldots, E_A(a_{N-1}), \theta_0, \ldots, \theta_{N-1}$ and Π.

1. Bob generates a sequence of values $\hat{a}_0, \ldots, \hat{a}_{N-1}$ where where $\hat{a}_i$ is chosen uniformly from $\{0, 1\}$ if $i \neq j$ and is 1 if $i = j$.
2. Bob generates a list of N items $\bar{\theta}_0, \ldots, \bar{\theta}_{N-1}$ where the jth value is 0 and all other values are chosen uniformly from $\mathbb{Z}^*_{M_B}$. He then creates a random permutation $\hat{\Pi}$ and permutes the values. Call this permuted list $\hat{\theta}_0, \ldots, \hat{\theta}_{N-1}$.
3. Bob outputs $E_A(\hat{a}_0), \ldots, E_A(\hat{a}_{N-1}), \hat{\theta}_0, \ldots, \hat{\theta}_{N-1}$ and $\hat{\Pi}$.

Figure 6.4. Algorithm S_A

contracts that are efficient in the sense that both parties may prefer another term. Therefore by efficient we mean Pareto-optimal: Any improvement for Alice must be at the expense of Bob and vice-versa. In this Section, we describe an extension that allows Alice and Bob to make preference choices through arbitrary utility functions that assign a desirability score to each term. We then filter out all terms that are not Pareto-optimal.

Let $U_A(x)$ (respectively, $U_B(x)$) denote Alice's (Bob's) utility for term x. In this Section we introduce a filtering protocol FILTER, that filters out inefficient solutions. We assume that any terms that are deemed unacceptable to a party have utility of 0 for that party, and we assume that all acceptable terms have unique utility (i.e, there are no ties). This last constraint is reasonable since if two terms have equal desirability, then the parties can easily just randomly assign unique utilities to each of the terms to break the ties.

Example 8 Returning to our example where $S = \{$London, New York, Chicago, Tokyo, Paris, Ottawa$\}$, $A = \{$London, New York, Paris, Ottawa$\}$ and $B = \{$London, Tokyo, Paris, Ottawa$\}$. Suppose Alice sets her utilities to $\{$London(3), New York(4), Chicago(0), Tokyo(0), Paris(1), Ottawa(2)$\}$, and Bob sets his utilities to $\{$London(3), New York(0), Chicago(0), Tokyo(1), Paris(4), Ottawa(2)$\}$. Recall that the original list of acceptable terms with utilities is $\{$London(3,3), Paris(1,4), Ottawa(2,2)$\}$. In this case Ottawa is an inefficient solution for this negotiation, because both parties prefer London to it. Thus the efficient terms are {London,Paris}.

It suffices to give a protocol for marking the terms of S that are inefficient. We do this by computing a value between Alice and Bob that is XOR-split (i.e., each party has a value, and the exclusive-or of their values is equal to the predicate "term is efficient"). It is a natural extension of the FIND-AGREEMENT protocol to utilize such values and we describe some of the details in Section 6.5.1. We omit a detailed proof of security, as this is a natural extension to the proofs outlined before. This filtering process is described in Figure 6.5.

Input: Alice has binary values $a_0, \ldots, a_{N-1}$, a set of integer utilities $A_0, \ldots, A_{N-1}$, a homomorphic encryption schemes E_A (where the modulus is M_A) and D_A. Bob also has a list of binary values $b_0, \ldots, b_{N-1}$, a set of integer utilities $B_0, \ldots, B_{N-1}$, and has E_A. It is also known that there is a term where $a_i \wedge b_i = 1$.
Output: Alice has binary values $\bar{a}_0, \ldots, \bar{a}_{N-1}$ and Bob has $\bar{b}_0, \ldots, \bar{b}_{N-1}$ where $\bar{a}_i \oplus \bar{b}_i = a_i \wedge b_i$ and the utility (A_i, B_i) is not dominated by another term. Furthermore, this list can be in any order.

1. Alice sends Bob $E_A(A_0), \ldots, E_A(A_{N-1})$.
2. For each i from 0 to $N-1$, Bob does the following:
   (a) Bob chooses a random values r_i in $\mathbb{Z}_{M_A}$.
   (b) If $b_i = 1$, then Bob computes $\alpha_i = E_A(a_i) * E_A(-r_i)$. And if $b_i = 0$, then Bob computes $\alpha_i = E_A(-r_i)$

   Bob sorts the α values in descending order according to his utility function. He sends these "sorted" values to Alice.
3. Alice decrypts these values to obtain c_i' and Bob sets $c_i'' = r_i$ (note that $c_i = A_{\Pi(i)}$ (for some permutation Π) if $a_{\Pi(i)} \wedge b_{\Pi(i)}$ and is 0 otherwise).
4. Alice and Bob engage in a Scrambled Circuit Evaluation that for each i from 0 to $N-1$, computes the maximum of the first $i-1$ items (for $i=0$ this is 0) and calls it m_i (which is stored in a split fashion). The circuit then sets $\bar{a}_i$ and $\bar{b}_i$ to $(m_i < c_i) \wedge (c_i \neq 0)$ in a XOR-split fashion. Clearly, this circuit can be constructed with $O(N)$ comparison circuits.

Figure 6.5. Protocol FILTER

6.5.1 Extending FIND-AGREEMENT

The only difference is that the protocol has to work on split values. We must show how Bob learns the values $E_A(a_i \wedge b_i)$. In this case Alice and Bob have values a_i and b_i where $a_i \oplus b_i$ is true iff both Alice and Bob find this term satisfiable and it is efficient. We now give a modified Step 1 and 2 of the FIND-AGREEMENT protocol to achieve this (we only describe things that are different):

1. Alice does the following: For each item a_i in the list $a_0, \ldots, a_{N-1}$, she creates pairs: $(E_A(a_i), E_A(\neg a_i))$. She sends these values to Bob.
2. Bob does the following: For each i from 0 to $N-1$, Bob chooses/computes: If $b_i = 0$, then he sets $\beta_i \leftarrow E_A(a_i) * E_A(0)$, and otherwise he sets it to $\beta_i \leftarrow E_A(\neg a_i) * E_A(0)$.

6.5.2 A Lower Bound on Communication Complexity

We now prove that finding a fair and efficient term has a communication complexity of $\Omega(N)$. We do this by showing a reduction from Set Disjointness (which has a lower bound of $\Omega(N)$ [64]). We now give a sketch of this proof:

Suppose Alice has a set A and Bob has a set B. Alice and Bob define another item (call it c) and both include it in their sets. They assign utilities to all items in their sets randomly, with the condition that the utility of c has to be lower than the utilities of all other items in their sets. They engage in a protocol to find a fair and efficient item. If the item is c, then the sets are disjoint and if the item is not c then the sets are not disjoint. □

6.6 Extensions

In this Section we outline three extensions. In Section 6.6.1 we discuss interactive negotiation. In Section 6.6.2 we discuss how to make our protocol's communication proportional to $O(|A| + |B|)$ (which could be more efficient than our previous solution). Finally, in Section 6.6.3 we outline how to handle dependent contract terms.

6.6.1 Interactive Negotiations

Consider what happens when the negotiators run the protocol and learn that no agreement is possible. If the parties abort the negotiation in this case, then this system may not be very useful for them. We now outline some strategies that will help the negotiation become more "interactive". We propose extending our previous framework to give feedback when there is no agreement possible. One of the problems with this approach is that the parties can perform a higher level of probing. We now explore various types of feedback; all of these types of feedback can easily be computed with simple circuits.

1. Alice and Bob learn how many clauses that are not satisfiable. If this number is low, then they could modify their values slightly in order to find an agreement, and if the number if high then they could break off the negotiation.

2. Alice and Bob learn if the number of clauses where there is no agreement is below some threshold. This clearly reveals less than the previous case, and it reveals similar information (i.e., it reveals when the parties are "close" to an agreement).

3. Alice and Bob could assign weights to each clause, and would learn if the weighted sum of the unsatisfiable clauses is below some threshold. It is possible that some terms are more difficult to agree upon than others, and thus this would allow Alice and Bob to have a more precise notion of what is "close".

4. Alice and Bob learn which clauses are not satisfiable. This reveals substantially more than any of the previous feedback techniques, but in this case Alice and Bob can target their changes.

5. Alice and Bob learn the clauses for which there is no agreement iff the number of terms (or weighted sum) is below some threshold. In this case, Alice and Bob only learn the information when they are close to an agreement. Thus nothing extra is revealed when the parties are far from an agreement.

6.6.2 Efficient Communication

The protocols outlined before our not particularly efficient if Alice and Bob's acceptability sets are much smaller than N. It would be desirable to have protocols with communication proportional to $|A| + |B|$. The downside to such a system is that it reveals "some" additional information, but we believe there are situations where such values are acceptable to leak. Our protocols can be modified to support such clauses, through usage of the protocols in [62]. We now give a high level view of how our protocols can be modified to achieve this.

To determine satisfiability, this is equivalent to determining if Alice and Bob's sets are disjoint, which can easily be computed with a circuit after the intersection is computed, and there is a protocol in [62] that allows such computations.

To compute a valid term we must determine which items in Alice's list are in Bob's list (in a XOR-split fashion, i.e., the result is split between them as in the protocol for finding and efficient agreement). Given such values, it is easy to compute a fair term using the protocol described in Section 6.5.1. This can easily be done by engaging in the set intersection protocol and then using an equality circuit, which is described in [62].

6.6.3 Dependent Contract Terms

In this Section we briefly outline an extension to our framework for dependent clauses. Two clauses are *dependent* if the value of one clause affects the acceptability set of another clause. For example, the location of a contract might affect which tasks a company is willing/capable to do. Another issue with dependency is if the dependency relationship is known globally or if it must be hidden. Here we assume that information about which clauses are dependent is public.

We now present a more formal definition of two-clause dependency (which can easily be generalized to n-clause dependency). Alice views clause C_2 as *dependent* on clause C_1 if the acceptability set for C_2 (call it A_2) is a function of the term chosen for C_1. Any contract with dependent clauses can be handled with our framework by taking every group of dependent clauses $C_1, \ldots, C_k$ and making a "super"-clause to represent all of the clauses. The set of states for this "super"-clause would be the k-tuples in the set $C_1 \times \cdots \times C_k$.

6.7 Summary

In this paper we define protocols for negotiating a contract between two entities without revealing their constraints for the contract. There are two essential issues that need to be addressed: i) is there an agreement for a contract and ii) if there is an agreement, then what is a valid, fair, and efficient contract. To provide efficiency we propose assigning utilities to terms and then filtering out inefficient solutions. To provide fairness the protocols choose a random efficient term in such a way that neither party has control over the choice of the term; the protocol for achieving fairness is the centerpiece of this exposition. Furthermore, we gave several extensions to our base protocols, including: feedback techniques, efficiency improvements for small sets, and protocols for dependent clauses.

7 BIOMETRICS

7.1 Introduction

Biometric-based identification starts with a physical measurement for capturing a user's biometric data, followed by the extraction of features from the measurement, and finally a comparison of the feature vector to some previously-stored reference vector. While biometric-based identification holds the promise of providing unforgeable authentication (because the biometric is physically attached to the user), it has a number of practical disadvantages. For example, the storage of reference vectors presents a serious privacy concern, since they usually contain sensitive information that many would prefer to keep private. Even from a security standpoint, biometric information must be stored and transmitted electronically, and, as the old adage goes, a user only gets nine chances to change her fingerprint password (and only one chance to change a retinal password). Thus, we would like to protect the privacy of biometric reference vectors.

One of the major difficulties with biometric information is that, even when it comes from the same individual, it is variable from one measurement to the next. This means that standard encryption of the reference vector is not sufficient to achieve the desired properties. For, even when the reference vector is stored in encrypted form, it appears as though the comparison step (comparing a recently-read biometric image to the reference vector) needs to be done in the clear. That is, standard techniques of comparing one-way hashes (or encryptions) of a stored password and an entered password cannot be used in the context of biometric authentication, as two very similar readings will produce very different hash (or encrypted) values. Unfortunately, this cleartext comparison of biometric data exposes sensitive information to capture by an adversary who obtains one of the two in-the-clear comparands, e.g., through spy-ware at the client or at the server. Moreover, in addition to this comparison-step vulnerability, encrypting the reference vector is obviously not sufficient to protect biometric data from an adversary who learns the decryption key, as could be the case with a dishonest insider at a financial institution. We next review previous work in overcoming these difficulties.

7.1.1 Related Work

There is a huge literature on biometric authentication, and we briefly focus here on the work most relevant to this Chapter. There are two broad approaches: The one where the comparison is done at the remote server, and the one where the comparison is done at the client end (the portable device where the biometric measurement is done). Most of the recent work has focused on the second category, ever since the landmark paper of Davida et al. [111] proposed that comparisons be done at the client end (although their scheme is also useful in the first case, of remote comparison at the server end). Many other papers (e.g., [112–114], to mention a few) build on the *wallet with observer* paradigm introduced in Chaum et al. [115] and much-used in the digital cash literature; it implies that there is a tamper-proof device available at the client's end where the comparison is made. The "approximate equality" in biometric comparisons is a challenge faced by any biometric scheme, and many ways have been proposed for overcoming that difficulty while preserving the required security properties. These include the use of error-correcting codes [111, 116–118], fuzzy commitments and fuzzy vaults [117, 119, 120] (for encrypting the private key on the smartcard using fingerprint information), fuzzy extractors [121], and the use of secure multi-party computation protocols [122]. Some carry out the comparisons "in the clear" (after decryption), whereas others manage to avoid it. They all rely, in varying degrees, to one (or more) of the following assumptions: That the portable device is tamper-resistant ("wallet with observer" based papers), that the portable device is powerful enough to carry out relatively expensive cryptographic computations (public-key encryption, homomorphic encryption), and sequences of these for carrying out complex multi-step protocols. See [123, 124] for a review and a general discussion of the pitfalls and perils of biometric authentication and identification (and how to avoid them), and [125] for a rather skeptical view of biometrics.

In spite of some drawbacks and practicality issues, these schemes have shown the theoretical (and, for some, practical) existence of secure and private biometric authentication.

7.1.2 Motivation for Our Approach

Just like the tiny (and weak) embedded microprocessors that are now pervasive in cars, machinery, and manufacturing plants, so will biometrically-enabled electro-mechanical devices follow a similar path to pervasiveness (this is already starting to happen due to security concerns). Not only inexpensive smartcards, but also small battery-operated sensors, embedded processors, and all kinds of other computationally weak and memory-limited devices may be called upon to carry out biometric authentication. Our work is based on the premise that biometric authentication will eventually be used in a such a pervasive manner, that it will be done on weak clients and servers, ones that can compute cryptographic hashes but not the more expensive cryptographic primitives and protocols; in a battery-powered device, this may be more for energy-consumption reasons than because the processor is slow. This paper explores the use of such inexpensive primitives, and shows that much can be achieved with them.

Our solutions have other desirable characteristics, such as not relying on physical tamper-resistance alone. We believe that relying entirely on tamper-resistance is a case of "putting too many eggs in one basket", just as would be a complete reliance on the assumed security of a remote online server – in either case there is a "single point of failure". See [126, 127] on the hazards of putting too much faith in tamper-resistance. It is desirable that a system's failure requires the compromise of *both* the client and remote server.

7.1.3 Lightweight Biometric Authentication

As stated above, we explore the use of lightweight computational primitives and simple protocols for carrying out secure biometric authentication. As in the previous schemes mentioned above, our security requirement is that an attacker should not learn the cleartext biometric data and should not be able to impersonate users by replaying encrypted (or otherwise disguised) data. Indeed, we would like a scheme to be resilient against insider attacks; that is, a (dishonest) insider should be unable to use data stored at the server to impersonate a user (even to the server). We also want our solutions to be simple and practical enough to be easily deployed on weak computational devices, especially at the client, which could be a small smartcard biometric reader. Even so, we don't want to rely on tamper-resistant hardware to store the reference vector at the client. Ideally, we desire solutions that make it infeasible for an attacker to impersonate a user even if the attacker steals the user's client device (e.g., a smartcard) and completely compromises its contents.

Even though the main rationale for this kind of investigation is that it makes possible the use of inexpensive portable units that are computationally weak (due to a slow processor, limited battery life, or both), it is always useful to provide such faster schemes even when powerful units are involved.

7.1.4 Our Contributions

The framework of our work is one where biometric measurement and feature extraction are done in a unit we henceforth refer to as the *reader*, which we sometimes refer to informally as the "smartcard," although this physical implementation of the reader is just one of many possibilities. It is assumed that the client has physical possession of the reader and, of course, the biometric itself. The alignment and comparison of the resulting measured feature information to the reference feature information is carried out at the *comparison unit*. Both the reference feature information and the comparison unit are assumed to be located at the *server* (at which authentication of the client is desired). Since authentication for financial transactions is a common application of biometric identification, we sometimes refer to the server informally as the "bank."

We present schemes for biometric authentication that can resist several possible attacks. In particular, we allow for the possibility of an attacker gaining access to the communication channel between reader and comparison unit, and/or somehow learning the reference information stored at the comparison unit (reference data is write-protected but could be read by insiders, spyware, etc.). We also allow for the possibility of an attacker stealing the reader from the client and learning the data stored on the reader. Such an attack will, of course, deny authentication service to the client, but it will not allow the attacker to impersonate the user, unless the attacker also obtains a cleartext biometric measurement from the user or the stored reference information at the server. To further resist even these two latter coordinated multiple attacks, the reader could have its data protected with tamper-resistant hardware, but we feel such coordinated multiple attacks (e.g., of simultaneously compromising the reader and the server) should be rare. Even so, tamper-resistant hardware protecting the memory at the reader could allow us to resist even such coordinated attacks.

Given such a rich mix of attacks that we wish to resist, it is desirable that the authentication protocol between reader and comparator not compromise the security or privacy of biometric information. We also require that compromise of the reference data in the comparator does not enable impersonation of the user. These security and privacy requirements pose a challenging problem because biometrics present the peculiar difficulty that the comparisons are necessarily inexact; they are for approximate equality. We give solutions that satisfy the following properties:

1. The protocols use cryptographic hash computations but not encryption. All the other operations used are inexpensive (no multiplication).

2. Information obtained by an eavesdropper during one round of authentication is useless for the next round, i.e., no replay attacks are possible.

3. User information obtained from the comparison unit by an adversary (e.g., through a corrupt insider or spyware) cannot be used to impersonate that user with that server or in any other context.

4. If a card is stolen and all its contents compromised, then the thief cannot impersonate the user.

Our solutions are based on a *decoupling* of information between the physical biometric, the reader, and the server, so that their communication and storage are protected and private, but the three of them can nevertheless perform robust biometric authentication. Moreover, each authentication in our scheme automatically sets up the parameters for the next authentication to also be performed securely and privately. Our scheme has the property that one smartcard is needed for each bank; this can be viewed as a drawback or as a feature, depending on the application at hand – a real bank is unlikely to trust a universal smartcard and will insist on its own, probably as an added security feature for its existing ATM card infrastructure. On the other hand, a universal card design for our framework of weak computational clients and servers (i.e., a card that works with many banks, as many of the above-mentioned earlier papers achieve) would be interesting and a worthwhile subject of further research. For now, our scheme should be viewed as a biometric supplement to, say, an ATM card's PIN; the PIN problem is trivial because the authentication test is of exact equality – our goal is to handle biometric data with the same efficiency and results as if a PIN had been used. Our scheme is not a competitor for the powerful PKI-like designs in the previous literature, but rather another point on a tradeoff between cost and performance.

We are not aware of any previous work that meets the above-mentioned security requirements using only lightweight primitives and protocols. The alignment stage, which precedes the comparison stage of biometric matching, need not involve any cryptographic computations even when security is a concern (cf. [122], which implemented a secure version of [128]). It is carrying out the (Hamming or more general) distance-computation in a secure manner that involves the expensive cryptographic primitives (in [122], homomorphic encryption). It is therefore on this “bottleneck” of the distance comparison that we henceforth focus, except that we do not restrict ourselves to Hamming distance and also consider other metrics.

7.2 Security Definition for Biometric Authentication

7.2.1 Adversary Model

An adversary is defined by the resources that it has. We now list these resources, and we also consider adversaries that have any combination of these resources:

1. *Smartcard (SCU and SCC):* An adversary may obtain an uncracked version of the client’s smartcard (SCU) or a cracked version of the smartcard (SCC). An adversary with SCU does not see the values on the smartcard, but can probe with various fingerprints. An adversary with SCC is also able to obtain all information on the smartcard. We consider an adversary that cracks the smartcard, modifies it, and then gives it back to the user to be outside of our adversary model.

2. *Fingerprint (FP):* An adversary may obtain someone’s fingerprint, by dusting for the print or by some more extreme measure.

3. *Eavesdrop (ESD, ECC, and ECU):* An adversary can eavesdrop on various components of the system. These include: i) The server’s database (ESD) which contains all information that the server stores about the client, ii) the communication channel (ECC) which has all information sent between the

client and server, and iii) the comparison unit (ECU) which has all information from ESD, ECC, and the result of the comparison.

4. *Malicious (MCC):* A stronger adversary would not only be able to eavesdrop on the communication channel, but could also modify values. We consider adversaries that can change the comparison unit or the server's database to be outside of our attack model.

7.2.2 Security Definitions

We look at the confidentiality, integrity, and availability of the system. The confidentiality requirements of the system are that an adversary should not be able to learn information about the fingerprint. The integrity of the system requires that an adversary cannot impersonate a client. The availability of the system requires that an adversary cannot make a client unable to login (i.e., "denial of service"). We now formally define the security requirements for the notions above.
Confidentiality:

We present three oracles that we consider acceptable (from a security standpoint), and we prove confidentiality by showing an adversary is equivalent to one of these oracles. In other words if given black-box access to such an oracle an adversary could emulate the real adversary's information, then the real adversary is "equivalent" to the oracle. We assume that the oracle has a copy of the "ideal" fingerprint $\bar{f}$.

1. Suppose the adversary has an oracle $A : \{0,1\}^{|\bar{f}|} \rightarrow \{0,1\}$, where $A(f)$ is true iff $\bar{f}$ and f are close. In other words, the adversary can try an arbitrary number of fingerprints and learn whether or not they are close to each other. We consider a protocol that allows such adversaries to be strongly secure.

2. Suppose the adversary has an oracle $B : \emptyset \rightarrow \{0,1\}^{\log |f|}$, where $B()$ returns the distance between several readings of a fingerprint (the actual fingerprints are unknown to the adversary). In other words, the adversary sees the distance

between several readings of a fingerprint. We consider a protocol that allows such adversaries to be strongly secure.

3. Suppose the adversary has an oracle $C : \{0,1\}^{|f|} \rightarrow \{0,1\}^{\log|\bar{f}|}$, where $C(f)$ returns the distance between $\bar{f}$ and f. In other words, the adversary can try many fingerprints and will learn the distance from the ideal fingerprint. Clearly, this adversary is stronger than the above mentioned adversaries. A protocol with such an adversary has acceptable security only in cases where the attack is detectable by the client; thus we call this weakly secure.

Integrity:

To ensure integrity we show that there is a check in place (either by the server or by the client) that an adversary with the specific resources cannot pass without having to invert a one-way function or guess a fingerprint. Of course if the adversary can weakly guess the fingerprint, then we say that the adversary can weakly impersonate the client.

Availability:

The types of denial of service attacks that we consider are those where the adversary can prevent the parties from communicating or can make the parties have inconsistent information which would make them unable to successfully authenticate.

7.2.3 Summary of Scheme's Security

Before we define the security of our system, we discuss the security (in the terms outlined above) of an "ideal" implementation. Such a system would require that the client use his fingerprint along with the smartcard and that all communication with the oracle take place through a secure communication channel. The user would be successfully authenticated to the server if and only if both the fingerprint and the smartcard were present. Clearly, we cannot do better than such an implementation.

Table 7.1 is a summary of an adversary's power with various resources (in our protocol); there are three categories of security: Strong, Weak, and No. The first two categories is defined in the previous Section, and "No" means that the system does not protect this resource against this type of adversary. Furthermore, we highlight the entries that are different from an "ideal" system. To avoid cluttering this exposition we do not enumerate all values in the table below, but rather for entries not in the table the adversary has capabilities equal to the maximum over all entries that it dominates.

Table 7.1
Security Properties for Biometric Scheme

Resources	**Confidentiality**	**Integrity**	**Availability**
FP	No	Strong	Strong
SCC and ESD	**No**	**No**	No
SCU and FP	No	No	No
MCC and ESD	Strong	**No**	No
SCU and ESD and MCC	**No**	**No**	No
MCC	Strong	Strong	No
SCU	Strong	Strong	No
SCU and ECU	**Weak**	**Weak**	No

Observe from the above table that the smartcard is the lynchpin of the system. While it is desirable to have a protocol that requires both the biometric and the smartcard, having the smartcard be the lynchpin is preferable to having the biometric be the lynchpin. The reason for this is that a biometric can be stolen without the theft being detected, however there is a physical trace when a smartcard is stolen (i.e., it is not there). The only exception to this is when the adversary has malicious control of the communication channel and can eavesdrop on the server's database, and in this case it can impersonate the client (but cannot learn the fingerprint).

7.3 Some False Starts

In this Section, we outline some preliminary protocols for biometric authentication that should be viewed as "warmups" for the better solutions given later in the paper. The purpose of giving preliminary protocols first is twofold: (*i*) to demonstrate the difficulty of this problem, and (*ii*) to provide insight into the protocol given later.

Initially, we give preliminary solutions for binary vectors and for the Hamming distance, however these preliminary solutions are extended to arbitrary vectors and other distance functions. The primary question that needs to be addressed is:

> "How does the bank compute the Hamming distance between two binary vectors without learning information about the vectors themselves?"

We assume that the server stores some information about some binary vector f_0 (the *reference vector*), and that the client sends the server some information about some other vector f_1 (the recently measured *biometric vector*). Furthermore, the server authenticates the client if $dist(f_0, f_1)$, the Hamming distance between f_0 and f_1, is below some threshold, ϵ. In addition to our security goal of being able to tolerate a number of possible attacks, there are two requirements for such a protocol:

- *Correctness:* the server should correctly compute $dist(f_0, f_1)$.
- *Privacy:* the protocol should reveal nothing about f_0 and f_1 other than the Hamming distance between the two vectors.

We now give various example protocols that attempt to achieve these goals, but nevertheless fail at some point:

1. Suppose the server stores f_0 and the client sends f_1 in the clear or encrypted for the server. This amounts to the naive (but common) solution mentioned above in the introduction. Clearly, this protocol satisfies the correctness property, but it does not satisfy the privacy requirement. In our architecture, this is vulnerable to insider attacks at the server and it reveals actual biometric data to the server.

2. Suppose, instead of storing f_0, the server stores $h(f_0||r)$, the result of a cryptographic one-way hash of f_0 and a random nonce, r. The client would then

need to compute $f_1||r$ and apply h to this string, sending the result, $h(f_1||r)$, to the server. This solution improves upon the previous protocol in that it protects the client's privacy. Indeed, the one-way property of the hash function, h, makes it computationally infeasible for the server to reconstruct f_0 given only $h(f_0||r)$. Unfortunately, this solution does not preserve the correctness of biometric authentication, since cryptographic hashing does not preserve the distance between objects. This scheme will work only for the case when $f_0 = f_1$, which is unlikely given the noise that is inherent in biometric measurements.

3. Suppose, then, that the server instead stores $f_0 \oplus r$ and the client sends $f_1 \oplus r$, for some random vector r known only to the client (where $\oplus$ is the component-wise XOR of the two vectors). This solution satisfies the correctness property for biometric authentication, because $dist(f_0 \oplus r, f_1 \oplus r) = dist(f_0, f_1)$ for the Hamming distance metric. This solution might at first seem to satisfy the privacy requirement, because it hides the number of 0's and 1's in the vectors f_0 and f_1. However, the server learns the positions where there is a difference between these vectors, which leaks information to the server with each authentication. This leakage is problematic, for after several authentication attempts the server will know statistics about the locations that differ frequently. Depending on the means of how feature vectors are extracted from the biometric, this leakage could reveal identifying characteristics of the client's biometric information. Thus, although it seems to be secure, this solution nonetheless violates the privacy constraint.

4. Suppose, therefore, that the scheme uses a more sophisticated obfuscating technique, requiring the server to store $\Pi(f_0 \oplus r)$, for some random vector r and some fixed random permutation (over the indices of the vector), Π, known only to the client. The client can authenticate in this case by sending $\Pi(f_1 \oplus r)$. This solution satisfies the correctness property, because $dist(\Pi(f_0 \oplus r), \Pi(f_1 \oplus r)) = dist(f_0, f_1)$, for the Hamming distance metric. Moreover, by using a random

permutation, the server does not learn the places in f_0 and f_1 where differences occur (just the places where the permuted vectors differ). Thus, for a single authentication round the server learns only the Hamming distance between f_0 and f_1. Unfortunately, this scheme nevertheless still leaks information with each authentication, since the server learns the places in the permuted vectors where they differ. Over time, because the same Π is used each time, this could allow the server to determine identifying information of the biometric.

This final scheme is clearly the most promising of the above false starts, in that it satisfies the correctness and privacy goals for a single authentication round. Our scheme for secure biometric authentication, in fact, is based on taking this final false start as a starting point for our actual protocol. The main challenge in making this scheme secure even for an arbitrarily long sequence of authentications is that we need a secure way of getting the server and client to agree on future permutations and random nonces (without violating the correctness and privacy constraints).

We now briefly discuss a variation of our final preliminary protocol for arbitrary vectors and other distance functions. Suppose that the reference vector (f_0) and the biometric vector (f_1) consist of elements chosen from $\mathbb{Z}_\Sigma$ and that the proximity decision is based on a distance function that depends on $|f_1 - f_0|$. The server stores $\Pi(f_0 + r)$ (addition modulo Σ) for some random vector r (with elements chosen from $\mathbb{Z}_\Sigma$) and some random permutation, Π, known only to the client. The client authenticates by sending $\Pi(f_1 + r)$. Clearly, this scheme suffers from the same drawbacks as the scheme for Hamming distance.

7.4 Our Schemes for Secure Biometric Authentication

In this Section, we give our protocols for secure biometric authentication. We begin with a protocol for the case of Boolean vectors where the relevant distance between two such vectors is the Hamming distance. We later extend this to vectors of arbitrary numbers and distance metrics that depend on differences between the corresponding components (this is a broad class that contains the Euclidean distance L_2, as well as L_1). We use $H(\cdot)$ to denote a keyed hash, where the key is a secret known to the client and server but not to others. An additional challenge in using such a function is that we now must prevent someone who accidentally (or maliciously) learns the client information at the server's end from using that information to impersonate the client to the server. Likewise, we must maintain the property that someone who learns the client's information on the reader should not be able to use this information (and possibly previously eavesdropped sessions) to impersonate the client.

7.4.1 Boolean Biometric Vectors

The server (in the database and the comparison unit) and the client (in the smartcard) store a small collection of values, which are recomputed after each round. Also, there are q copies of this information at the server and on the card, where q is the number of fingerprint mismatches before a person must manually re-register with the server. In what follows, f_i and f_{i+1} are Boolean vectors derived from biometric readings at the client's end, Π_i and Π_{i+1} denote random permutations generated by and known to the client but not the server, and $r_i, r_{i+1}, s_i, s_{i+1}, s_{i+2}$ are random Boolean vectors generated by the client, some of which may end up being revealed to the server.

Before a round, the server and client store the following values:

- The server has: $s_i \oplus \Pi_i(f_i \oplus r_i)$, $H(s_i)$, $H(s_i, H(s_{i+1}))$.
- The client has: Π_i, r_i, s_i, s_{i+1}.

A round of authentication must not only convince the server that the client has a vector f_{i+1} that is "close" (in the Hamming distance sense) to f_i, but must also refresh the above information. Figure 7.1 shows what a round of the protocol consists of:

1. The client uses the smartcard to read a new biometric f_{i+1} and to generate random Boolean vectors r_{i+1} and s_{i+2} and a random permutation Π_{i+1}.
2. The smartcard connects to the terminal and sends to the server the following values: $\Pi_i(f_{i+1} \oplus r_i)$, s_i, and "transaction information" T that consists of a nonce as well as some other information related to this particular access request (e.g., date and time, etc).
3. The server computes the hash of the just-received s_i and checks that it is equal to the previously-stored $H(s_i)$. If this check does not match it aborts the protocol. If it does match, then the server computes the XOR of s_i with the previously-stored $s_i \oplus \Pi_i(f_i \oplus r_i)$ and obtains $\Pi_i(f_i \oplus r_i)$. Then the server computes the Hamming distance between the just-computed $\Pi_i(f_i \oplus r_i)$ and the received $\Pi_i(f_{i+1} \oplus r_i)$.
   - If the outcome is a match, then the server sends $H(T)$ to the client.
   - If it is not a match, then the server aborts but throws away this set of information in order to prevent replay attacks; if the server does not have any more authentication parts, then it locks the account and requires the client to re-register.
4. The smartcard checks that the value sent back from the server matches $H(T)$ (recall that H is a keyed hash). If the message does not match, the smartcard sends an error to the server. Otherwise, the smartcard sends the server the following information: $s_{i+1} \oplus \Pi_{i+1}(f_{i+1} \oplus r_{i+1})$, $H(s_{i+1}, H(s_{i+2}))$, and $H(s_{i+1})$. It also wipes from its memory the reading of fingerprint f_{i+1} and of previous random values r_i and s_i, so it is left with Π_{i+1}, r_{i+1}, s_{i+1}, s_{i+2}.
5. When the server receives this message it verifies that $H(s_i, H(s_{i+1}))$ matches the previous value that it has for this quantity and then updates its stored values to: $s_{i+1} \oplus \Pi_{i+1}(f_{i+1} \oplus r_{i+1})$, $H(s_{i+1}, H(s_{i+2}))$, and $H(s_{i+1})$.

Figure 7.1. Protocol for Biometric Authentication

7.4.2 Arbitrary Biometric Vectors

Suppose the biometric vectors f_i and f_{i+1} now contain arbitrary (rather than binary) values, and the proximity decision is based on a distance function that depends on $|f_i - f_{i+1}|$. To extend our protocol to such distances, all that needs to be done is to modify the description of the Boolean protocol as follows:

- Each of r_i, r_{i+1} is now a vector of arbitrary numerical values rather than Boolean values (but s_i, s_{i+1}, s_{i+2} are still Boolean).

- Every $f_j \oplus x$ gets replaced in the protocol's description by $f_j + x$, e.g., $f_i \oplus r_i$ becomes $f_i + r_i$. (The length of s_i must of course now be the same as the number of bits in the binary representation of $f_i + r_i$, but we refrain from belaboring this straightforward issue.)

The above requires communication $O((\log|\Sigma|)n)$, where Σ is the size of the alphabet and n is the number of items. This reveals slightly more than the distance, in that it reveals the component-wise differences. This information leakage is minimal especially since the values are permuted, but clearly a protocol that does not leak such information is preferred. In the case where the function is $\sum_{i=1}^{n}|f_i - f_{i+1}|$, then by using a unary encoding for each value this reduces to a Hamming distance computation, for which the protocols of the previous Section can be used. This does not reveal the component-wise differences, but it requires $O(|\Sigma|n)$ communication.

7.5 Security of the Protocols

In this Section, we prove that Table 7.1 in Section 7.2.3 is accurate. First, we define the information and abilities of the adversaries, and the prove the confidentiality, integrity and availability constraints.

7.5.1 Adversary Resources

Table 7.2 summarizes the information available to various adversaries. Generally, an adversary with multiple resources gets all of the information of each resource. There are cases where this is not the case, e.g., consider an adversary with SCU and ECC; the adversary can't see readings of the client's fingerprint, because the client no longer has the smartcard and thus is not able to attempt an authentication protocol.

Table 7.2
Resources for Various Adversaries

Adversary	**Information**
FP	f
SCU	Ability to probe small number of fingerprints
SCC	SCU and r_i, s_i, Π_i, k
ESD	k and several sets of $H(s_i), H(s_i, H(s_{i+1})), s_i \oplus \Pi_i(f \oplus r_i)$
ECC	Several sets of $s_i, \Pi_i(f \oplus r_i), H(s_{i+1}), H(s_{i+2})$
ECU	ESD and ECC and distances of several readings
MCC	ECC and can change values

7.5.2 Proof of Confidentiality

Before we prove the confidentiality requirements we need the following lemma:

Lemma 5 *The pair of values* $(\Pi(f \oplus r))$ *and* $(\Pi(f' \oplus r))$ *reveals nothing other than the distance between each pair of vectors* f *and* f'.

Proof: Suppose we are given a specific distance d (which is the output of the protocol). We must show that for any pair of vectors (x, x') with distance d that there are equally many $(\bar{\Pi}, \bar{r})$ pairs such that $\bar{\Pi}(x \oplus \bar{r}) = (\Pi(f \oplus r))$ and $\bar{\Pi}(x' \oplus \bar{r}) = (\Pi(f' \oplus r))$. By showing this, we will have shown that the values reveal nothing other than what can be simulated from their distance (in an information theoretic sense). Since the distance is d in both cases there will be exactly d locations where the vectors x and x' differ. Any permutation $\bar{\Pi}$ that maps the positions that x and x' differ to the positions that f and f' differ will create two vectors that disagree at the same positions as f and f', and there are an equal number of such permutations. For each such permutation there is a unique vector $\bar{r}$ such that the values match, and thus the number of such $(\bar{\Pi}, \bar{r})$ pairs is the same regardless of the vectors (x, x').

$\square$

Theorem 7.5.1 *The only cases where an adversary learns the fingerprint are in: i) FP, ii) SCC and ESD, iii) SCU and ESD and MCC, and iv) any superset of these cases. In the case of SCU and ECU the adversary weakly learns the fingerprint.*

Proof: First when the adversary has the fingerprint the theorem is clearly true. Suppose that the adversary has ECU and MCC, the adversary sees several pairs of $\Pi(f \oplus r)$ and $\Pi(f' \oplus r)$ and by Lemma 1, this only reveals a set of distances, which is equivalent to oracle B and thus is secure. Thus any attack must involve an adversary with the smartcard in some form. Clearly, any adversary with the smartcard cannot eavesdrop on communication when the client is logging into the system.

Suppose that the adversary has SCC and MCC. The adversary has no information about the fingerprint in any of its information, and since nothing is on the smartcard and a client cannot login without the smartcard, the fingerprint is protected. However, if the adversary has SCC and ESD, they can trivially learn the fingerprint from knowing $\Pi_i, r_i, s_i, s_i \oplus \Pi_i(f \oplus r_i)$.

Any adversary with SCU can only probe various fingerprints, as no other information is given. Suppose that the adversary has SCU and ECU. In this case the adversary can probe various fingerprints and can learn the distance, which is equivalent to oracle C and thus is weakly secure. Consider an adversary with SCU and ESD. In this case it can probe using the SCU, but this is just oracle A. If the adversary has SCU and MCC, then it can learn s, Π, and r values by stopping the communication and trying various fingerprints, however the adversary cannot use this to glean the fingerprint as the client cannot login once the smartcard is stolen. Finally, if the adversary has SCU and MCC and ESD, then it can learn the values s, Π, and r and then use the information from ESD to learn the fingerprint. $\square$

7.5.3 Proof of Integrity

Theorem 7.5.2 *The only cases where an adversary can impersonate a client are in: i) SCU+FP, ii) SCC and ESD, iii) MCC and ESD, and iv) any superset of these cases. In the case of SCU and ECU the adversary weakly impersonate the client.*

Proof: We first show that the above mentioned adversaries can impersonate the client. Clearly an adversary with SCU and FP can trivially impersonate the client. An adversary with SCC and ESD can learn the fingerprint, and again it can trivially impersonate the client. An adversary with MCC and ESD could perform a man in the middle attack, since it knows k it can impersonate the server and then perform a replay attack with the actual server. SCU and ECU can weakly learn the fingerprint and thus can trivially weakly impersonate the client. It is limited to only weak impersonation, because if not this would imply that the adversary could learn the fingerprint, which contradicts Theorem 7.5.1.

Now we must show that any weaker adversary cannot impersonate the client, we do this by showing that adversaries with i) MCC and FP, ii) ECU and ECC and FP, iii) SCC and MCC, and iv) SCU and ESD and ECC cannot impersonate the client. We now enumerate these cases:

1. *MCC and FP*: Any attack is prevented, because on Step 4 of the protocol where the smartcard checks if the transaction was hashed with k correctly. This adversary does not know k and thus cannot generate such a message. If the protocol has made it this far, then the protocol will not reuse the same value of s again.
2. *ECU and ECC and FP*: Since this adversary is passive it cannot disrupt the flow of any transaction, and thus once a value of s is used it is never used again. This adversary cannot guess the next value of s without inverting a one-way function.
3. *SCC and MCC*: The malicious component does little here, since the client cannot attempt to login when the adversary has the smartcard. And since the smartcard contains no information about f this adversary cannot guess the fingerprint.
4. *SCU and ESD and ECC*: The adversary cannot listen in on an active transaction as it has the smartcard. Also, since it is passive it cannot disrupt the flow of the login process. The information the adversary has does not allow it to gain the fingerprint, and thus it cannot impersonate the client.

$\square$

7.5.4 Proof of Availability

Theorem 7.5.3 *The only cases where an adversary can attack the availability of the client are in: i) SCU, ii) MCC, and any superset of these cases.*

Clearly any adversary with the smartcard in any form can stop availability, as the client no longer has the smartcard. Furthermore, any adversary that can maliciously control the communication channel can easily prevent the user from sending the authentication information. To show that no other adversary can mount such an attack, we consider an adversary with FP and ECU (which is the most powerful adversary that does not have the smartcard or malicious control of the communication channel). The only way to make the smartcard and the server's information out of sync is to know the value s_i, but this is as hard as inverting a one-way function. $\square$

7.6 Storage-Computation Tradeoff

In this Section, we introduce a protocol that allows q fingerprint mismatches before requiring the client to re-register with the server, with only $O(1)$ storage, but that requires $O(q)$ hashes to authenticate. This utilizes similar ideas as SKEY [7]; in what follows $H^j(x)$ denotes the value of x hashed j times. We do not prove the security of this system as it is a natural extension to the previous protocol. After the setup the following is the state of the system:

- Server has: $\bigoplus_{j=0}^{q-1} H^j(s_i) \oplus \Pi_i(f_i \oplus r_i)$, $H^q(s_i)$, and $H(H^q(s_i), H^q(s_{i+1}))$.
- Client has: Π_i, r_i, s_i, and s_{i+1}.

After t fingerprint mismatches the server has: $\bigoplus_{j=0}^{q-t-1} H^j(s_i) \oplus \Pi_i(f_i \oplus r_i)$, $H^{q-t}(s_i)$, and $H(H^q(s_i), H^q(s_{i+1}))$.

The authentication and information-updating round is as follows for the tth attempt to authenticate the client is described in Figure 7.2.

1. The client uses the smartcard to read a new biometric f_{i+1} and to generate random Boolean vectors r_{i+1} and s_{i+2} and a random permutation Π_{i+1}.
2. The smartcard connects to the terminal and sends to the server the following values: $\bigoplus_{j=0}^{q-t-1} H^j(s_i) \oplus \Pi_i(f_{i+1} \oplus r_i)$ and $H^{q-t}(s_i)$.
3. The server computes the hash of the just-received $H^{q-t}(s_i)$ and checks that it is equal to the previously-stored $H^{q-t+1}(s_i)$. If this check does not match it aborts the protocol. If it does match, then the server computes the XOR of $H^{q-t}(s_i)$ with the previously-stored $\bigoplus_{j=0}^{q-t} H^j(s_i) \oplus \Pi_i(f_i \oplus r_i)$ and obtains $\bigoplus_{j=0}^{q-t-1} H^j(s_i) \oplus \Pi_i(f_i \oplus r_i)$. It then computes the Hamming distance between the just-computed $\bigoplus_{j=0}^{q-t-1} H^j(s_i) \oplus \Pi_i(f_i \oplus r_i)$ and the received $\bigoplus_{j=0}^{q-t-1} H^j(s_i) \oplus \Pi_i(f_{i+1} \oplus r_i)$.
   - If the outcome is a match, then the server sends $H(T)$ (recall that H is a keyed hash) to the client.
   - If it is not a match, then the server updates its values to the following: $\bigoplus_{j=0}^{q-t-1} H^j(s_i) \oplus \Pi_i(f_i \oplus r_i)$, $H^{q-t}(s_i)$, and $H(H^q(s_i), H^q(s_{i+1}))$. If $t = q$, then the server locks the account and requires the client to re-register.
4. The smartcard checks that the value sent back from the server matches $H(T)$, and if it is a match then the smartcard sends the server the following information: $\bigoplus_{j=0}^{q-1} H^j(s_{i+1}) \oplus \Pi_{i+1}(f_{i+1} \oplus r_{i+1})$, as well as $H^q(s_{i+1})$, and also $H(H^q(s_{i+1}), H^q(s_{i+2}))$. If it does not match, then it sends an error to the server and aborts. In either case, it wipes from its memory the reading of fingerprint f_{i+1} and those previously stored values that are no longer relevant.
5. When the server receives this message it verifies that $H(H^q(s_i), H^q(s_{i+1}))$ matches the previous value that it has for this quantity and then updates its stored values to: $\bigoplus_{j=0}^{q-1} H^j(s_{i+1}) \oplus \Pi_{i+1}(f_{i+1} \oplus r_{i+1})$, $H^q(s_{i+1})$, and $H(H^q(s_{i+1}), H^q(s_{i+2}))$.

Figure 7.2. Storage-Computation Tradeoff for Biometric Authentication

7.7 Conclusions and Future Work

In this Chapter, a lightweight scheme was introduced for biometric authentication that could be used by weak computational devices. Unlike other protocols for this problem, our solution does not require complex cryptographic primitives, but instead relies on cryptographic hashes. Our protocols are secure in that the client's fingerprint is protected, i.e., it is "hard" to impersonate a client to the comparison unit, and adversaries with malicious access to the communication channel cannot steal a client's identity (i.e., be able to impersonate the client to the comparison unit after the transaction). To be more precise, an adversary would need the smartcard and either the fingerprint or the server's database to impersonate the client.

8 SUMMARY

In this thesis we have introduced privacy and confidentiality preserving protocols for problems in a number of domains. These protocols are more efficient and simple than the protocols implied by the earlier general constructions. The domains explored by this thesis include: trust negotiation, credit checking, services for location-aware devices, contract negotiation, and secure biometric authentication. While these domains may appear to be unrelated other than the privacy-preserving framework, the significance and applicability of this work is not limited to these specific domains, as these domains are each a "prototype" for a class of similar applications. The rest of this Chapter is organized as follows. In Section 8.1 we describe the main results of this thesis in more detail, and in Section 8.2 we describe future work.

8.1 Summary of Main Results

In this Section we outline the major contributions of this thesis:

1. *Comparing Modularly split values and Base Reduction* (Sections 3.7.1 and 3.7.2): Previously, when values were split amongst two entities and needed to be compared, the values were additively split without using modular arithmetic. The problem with this is that this probabilistically leaks information when the randomness is chosen near a boundary (i.e., if a non-negative value x is hidden by adding a random value r in the range $[0, R]$ and the value $x + r$ is 0, then one can deduce that x is 0). And while the probability of a leak can be made arbitrary small, it is better to avoid this problem. We showed that it is possible to compare modular additively split values (which do not have the above-mentioned problem) under certain reasonable constraints with similar costs to a standard secure comparison. Furthermore, we introduced

base reduction which allows modularly split values to shrink to the minimum required size (rather than the size of a security parameter).

2. *Hidden Credentials and Hidden Access Policies* (Section 4.4): We introduced protocols that allow a client to access materials without revealing his credentials and without learning the server's access policies. Part of this is achieved by using secure oblivious circuit evaluation, but another component is verifying that the client actually has credentials. This is complicated by the fact that the verification authority cannot be an active part of the process. We introduced techniques that achieve this verification based upon Identity Based Encryption, Set Intersection, and Scrambled Circuit Evaluation.

3. *Credit Checking:* (Section 4.5): We investigated a new approach to credit checking that allows the borrower to protect his credit report yet assures the lender that the report is accurate. This is done with a three party protocol between the borrower, the lender, and the credit report agency. The communication architecture is almost identical to the current system of non-private credit checking, and thus the CRA's only task in the protocol is to send the credit report in a garbled form to the lender. The borrower and the lender then engage in a protocol to determine loan qualification. Furthermore, this is achieved without requiring expensive cryptographic operations (nothing more than symmetric key cryptography).

4. *Services for Location-Aware Devices* (Chapter 5): We introduced several protocols for computing various distance functions. The applications for these techniques include: finding the nearest point of interest to a route, determining if a route is "safe" or "desirable", and determining if two routes are too close. We also investigated efficient techniques based upon data perturbation. Many of these techniques do not require the server to run a special protocol; thus these techniques could be used even if the server does not agree to run a special protocol, but rather just answers nearest neighbor queries in a non-

private fashion. Some of these techniques have an accuracy-privacy tradeoff, and others have a communication-privacy tradeoff, but they are substantially more efficient than the perfectly secure protocols for these same problems.

5. *Contract Negotiation* (Chapter 6): We introduced a framework for two-party contract negotiation. In this framework the users first securely determine if an agreement is possible: If not then the negotiators will not engage in other protocols with their current constraints, but if they learn that a valid contract exists then they securely compute a valid, fair (i.e., one where neither party controls the outcome) and efficient (i.e., one where no clause is replaceable by another that is better for both parties) contract. We introduced protocols for achieving the above that are more efficient than the solutions implied by the general results of SMC.

6. *Biometrics* (Chapter 7): We introduced a simple biometric comparison protocol that uses only lightweight cryptography. One of the problems with using biometrics for authentication is that once a biometric is compromised, it is difficult to change the biometric (it is not desirable to use compromised biometrics). Thus a simple authentication protocol that reveals the actual biometrics (or features in them) during the authentication is highly risky. Furthermore, techniques for protecting passwords (such as hashing) do not work with biometrics, as they are inherently noisy. Thus the goal is to have a biometric authentication system where the server knows that two biometrics are close without having to know the actual biometrics. We gave lightweight protocols for achieving the above task.

In summary, we have introduced many new techniques for secure protocols and have applied them to a number of application domains that cover a broad range of online interactions. We expect that many of these techniques will be useful in other domains.

8.2 Future Work

In this Section we introduce future work for secure and private online collaboration. In Section 8.2.1 we introduce many interesting problems for specific domains, and in Section 8.2.2 we describe larger challenges in this area.

8.2.1 Domain Specific Future Work

In this Section we outline specific future work for the domains studied in this thesis.

Trust Negotiation

While our scheme does not require that the policy be monotonic (i.e., it can require the absence of a credential), this feature would not be used for practical reasons (as it is hard to enforce a requirement for the absence of a credential). A variation on such a scheme is to allow the CA to issue groups of credentials where the client can use them in an "all or none" fashion. This is similar to a driver's license where you cannot use it to give your height without revealing your age. The advantage of such a scheme is that the client would have to use all of his/her credentials to use any of them, which could give the client an incentive to use negative credentials. Thereby, enabling a need to support non-monotonic policies.

Another issue in privacy-preserving trust negotiation is: While our protocols do not leak information to the server about whether or not a client successfully obtained access to a message, there are cases where this information is leaked (regardless of which of out protocols is used). As an example, suppose that the server grants access to services rather than messages. For some types of services the server cannot grant access without learning if the client should obtain access to the service. As another example, suppose that message requests are dependent on the outcome of previous requests, e.g., suppose that a client requests message M_1, which contains a hyperlink to message M_2, and then that same client requests M_2 a few minutes later. In this case, the server does not learn for certain that the client successfully obtained M_1, but inferences can be made. Thus, in our scheme for hiding credentials and policies there is potential for information leakage; that is the server can probe the client's credential sets (at least probabilistically). This is particularly damaging when the gain from learning when a client has a specific credential is more than the cost of giving a message or service to a non-deserving client, because the server can then set the policy to be a single credential. This is bad because the client will not even be able to detect that such probing behavior has occurred. In summary, our scheme does not adequately protect the client's credentials against probing by the server; we call this problem the *Sensitive CredentiAl Leakage Problem (SCALP).* We reiterate that SCALP is not due to any specific weaknesses in our protocols, but rather exists in any system where the server can link transactions to the same client and has arbitrary freedom about creating the access policy. Since such leakages are unavoidable in this system, the client may have legitimate concerns about using credentials that he deems sensitive. Thus when the client is told to protect certain credentials (e.g., a "top secret clearance" credential), then the client should not use this credential in our scheme. Clearly, this is problematic for any negotiation system. A strategy to partially resolve SCALP would be to allow the client and the server to input access policies for their credentials; thus they could control the situations in which their credentials may be leaked. Another approach would be to require some third party off-line verification of the access policy (perhaps by a policy authority), which would make probing attacks more difficult.

Services for Location-Aware Devices

One direction for future work is computing other distance functions such as distance between parametric lines with acceleration. Another direction is creating protocols for a route planner. For example, given a set of points with threshold distances and a start and end point, compute a route from the start point to the end point without violating the distance thresholds.

There are also many future directions for perturbation-based techniques. For example, creating perturbation-based techniques and analyzing their error in situations where the computation is not limited to functions on points (for example a scheme that perturbs routes). Another direction is to investigate the applicability of perturbation techniques for determining whether two objects are too close.

Contract Negotiation

Possible future work includes: i) protocols for dependent clauses that are more efficient than the generic equivalents, ii) protocols for specific terms that are more efficient than the generic protocols presented in this paper, iii) extending the framework to more than two parties, iv) extending the protocols to a model of adversary besides semi-honest, and v) extending the framework to allow multiple negotiations with inter-contract dependencies.

Biometrics

One problem with our protocol is that, for every successful authentication, the database must update its entry to a new value (to prevent replay attacks), and thus we pose the following open problem: is it possible for the server to have a static database and have a secure authentication mechanism requiring only cryptographic hash functions?

Other Domains

A natural avenue for future research is simultaneously handling all of the facets of online collaboration that have been studied in this thesis (certified inputs, multiple outcomes, etc). Furthermore, another area of future work is identifying meaningful new application domains where secure protocols are useful, and to develop protocols for these domains.

8.2.2 Greater Challenges

In this Section we describe larger challenges in the domain of secure protocols.

1. *Implementation:* Implementation of many secure protocol techniques would be very useful. While the general results about secure protocols are quite voluminous, there is little in the form of implementations. Thus, in order to understand what is tractable, and to learn where improvements must be made, an implementation of many such protocols would be a springboard to many new research problems.

2. *Adversary models:* Another challenge is in defining new adversary models. Specifically, developing models that are stronger than the semi-honest model but not as strong as the malicious model. The malicious model is a very strong notion in that it allows the adversaries to be arbitrary PPT algorithms. There may be many cases where the full malicious model is not needed, perhaps due to the incentives of the players. An example of such a adversary model is: It is possible for the adversary to achieve some effect that is not achievable in the ideal model, but the adversary is no better off then if he engaged in the protocol truthfully. This would be particularly interesting if the protocol for the weaker model is more efficient than the protocol for the malicious model.

3. *Weaker Notions of Privacy:* Another problem with secure protocols is the notion that nothing can be leaked. This is clearly a desirable goal, but we there may be many cases where this is too restrictive. This would be particularly interesting if by allowing some information to be leaked that one could create a protocol that is more efficient than a non-leaky protocol. The difficulty with

this approach is defining what “acceptable” losses are, and then creating a security model that captures this notion.

4. *Weaker notion of Robustness:* Most of the multi-party protocols require robustness against Byzantine adversaries. Furthermore, resilience against these adversaries has been achieved by the general results in an optimal sense, however this full degree of resilience may not be necessary. It would be interesting to explore weaker notions of robustness to allow the development of more efficient protocols.

LIST OF REFERENCES

LIST OF REFERENCES

[1] O. Goldreich. Cryptography and cryptographic protocols. *Journal of Distributed Computing*, 16(2-3):177–199, 2003.

[2] O. Goldreich. *Foundations of Cryptography: Volume II Basic Application*. Cambridge University Press, 2004.

[3] S. Goldwasser. Multi-party computations: Past and present. In *Proceedings of the Sixteenth Annual ACM Symposium on Principles of Distributed Computing*, pages 1–6. ACM Press, 1997.

[4] S. Micali and P. Rogaway. Secure computation (abstract). *Advances of Cryptology (CRYPTO 1991)*, LNCS 576:392–404, 1992.

[5] D. Beaver. Foundations of secure interactive computing. *Advances in Cryptology (CRYPTO 1991)*, LNCS 576:377–391, 1992.

[6] P. Feldman and S. Micali. Optimal algorithms for byzantine agreement. In *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, pages 148–161. ACM Press, 1988.

[7] B. Schneier. *Applied Cryptography: Protocols, Algorithms, and Source Code in C*. John Wiley & Sons, Inc., 2nd edition, 1995.

[8] O. Goldreich. *Foundations of Cryptography: Volume I Basic Tools*. Cambridge University Press, 2001.

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

[10] A.C Yao. How to generate and exchange secrets. In *Proceedings of the 27th Annual IEEE Symposium on Foundations of Computer Science*, pages 162–167, 1986.

[11] D. Malkhi, N. Nisan, B. Pinkas, and Y. Sella. Fairplay – A secure two-party computation system. In *Proceedings of Usenix Security*, 2004.

[12] C. Cachin, J. Camenisch, J. Kilian, and J. Müller. One-round secure computation and secure autonomous mobile agents. *Proceedings of Twenty-seventh International Colloquium on Automata, Languages and Programming*, LNCS 1853, 2000.

[13] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game. In *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing*, pages 218–229. ACM Press, 1987.

[14] Y. Lindell. Parallel coin-tossing and constant-round secure two-party computation. *Advances in Cryptology (CRYPTO 2001)*, LNCS 2139:171–189, 2001.

[15] J. Katz and R. Ostrovsky. Round optimal secure two-party computation. *Advances in Cryptology (CRYPTO 2004)*, LNCS 3152, 2004.

[16] M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, pages 1–10. ACM Press, 1988.

[17] D. Chaum, C. Crépeau, and I. Damgård. Multiparty unconditionally secure protocols. In *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, pages 11–19. ACM Press, 1988.

[18] J. Bar-Ilan and D. Beaver. Non-cryptographic fault-tolerant computing in constant number of rounds of interaction. In *Proceedings of the Eighth Annual ACM Symposium on Principles of Distributed Computing*, pages 201–209. ACM Press, 1989.

[19] D. Beaver, S. Micali, and P. Rogaway. The round complexity of secure protocols. In *Proceedings of the Twenty-second Annual ACM Symposium on Theory of Computing*, pages 503–513. ACM Press, 1990.

[20] T. Rabin and M. Ben-Or. Verifiable secret sharing and multiparty protocols with honest majority. In *Proceedings of the Twenty-first Annual ACM Symposium on Theory of Computing*, pages 73–85. ACM Press, 1989.

[21] D. Beaver. Multiparty protocols tolerating half faulty processors. *Advances in Cryptology (CRYPTO 1989)*, LNCS 435:560–572, 1990.

[22] R. Gennaro, M.O. Rabin, and T. Rabin. Simplified vss and fast-track multiparty computations with applications to threshold cryptography. In *Proceedings of the Seventeenth Annual ACM Symposium on Principles of Distributed Computing*, pages 101–111. ACM Press, 1998.

[23] R. Gennaro, Y. Ishai, E. Kushilevitz, and T. Rabin. The round complexity of verifiable secret sharing and secure multicast. In *Proceedings of the Thirty-third Annual ACM Symposium on Theory of Computing*, pages 580–589. ACM Press, 2001.

[24] R. Cramer and I. Damgård. Secure distributed linear algebra in a constant number of rounds. *Advances in Cryptology (CRYPTO 2001)*, LNCS 2139:119–136, 2001.

[25] R. Cramer, I. Damgård, and J. Nielsen. Multiparty computation from threshold homomorphic encryption. *Advances in Cryptology (EUROCRYPT 2001)*, LNCS 2045:280–300, 2001.

[26] M. Jakobsson and A. Juels. Secure function evaluation via ciphertexts. *Advances in Cryptology (ASIACRYPT 2000)*, LNCS 1976:162–173, 2000.

[27] Y. Ishai and E. Kushilevitz. Randomizing polynomials: A new representation with applications to round-efficient secure computation. In *41st Annual Symposium on Foundations of Computer Science, 2000*, pages 294–304. IEEE, 2000.

[28] R. Cramer, S. Fehr, Y. Ishai, and E. Kushilevitz. Efficient multi-party computation over rings. *Advances in Cryptology (EUROCRYPT 2003)*, LNCS 2656:596–613, 2003.

[29] J. Katz, R. Ostrovsky, and A. Smith. Round efficiency of multi-party computation with a dishonest majority. *Advances in Cryptology (EUROCRYPT 2003)*, LNCS 2656:578–595, 2003.

[30] M. Ben-Or, R. Canetti, and O. Goldreich. Asynchronous secure computation. In *Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing*, pages 52–61. ACM Press, 1993.

[31] M. Ben-Or, B. Kelmer, and T. Rabin. Asynchronous secure computations with optimal resilience (extended abstract). In *Proceedings of the Thirteenth Annual ACM Symposium on Principles of Distributed Computing*, pages 183–192. ACM Press, 1994.

[32] R. Canetti, U. Feige, O. Goldreich, and M. Naor. Adaptively secure multi-party computation. In *Proceedings of the Twenty-eighth Annual ACM Symposium on Theory of Computing*, pages 639–648. ACM Press, 1996.

[33] R. Cramer, I. Damgård, S. Dziembowski, M. Hirt, and T. Rabin. Efficient multiparty computations secure against an adaptive adversary. *Advances in Cryptology (EUROCRYPT 1999)*, LNCS 1592:311–326, 1999.

[34] M. Hirt and U. Maurer. Complete characterization of adversaries tolerable in secure multi-party computation (extended abstract). In *Proceedings of the Sixteenth Annual ACM Symposium on Principles of Distributed Computing*, pages 25–34. ACM Press, 1997.

[35] C. Crépeau, D. Gottesman, and A. Smith. Secure multi-party quantum computation. In *Proceedings of the Thirty-fourth Annual ACM Symposium on Theory of Computing*, pages 643–652, New York, NY, USA, 2002. ACM Press.

[36] U. Feige, J. Killian, and M. Naor. A minimal model for secure computation (extended abstract). In *Proceedings of the Twenty-sixth Annual ACM Symposium on Theory of Computing*, pages 554–563. ACM Press, 1994.

[37] D. Beaver. Commodity-based cryptography (extended abstract). In *Proceedings of the Twenty-ninth Annual ACM Symposium on Theory of Computing*, pages 446–455. ACM Press, 1997.

[38] T. Sander, A. Young, and Moti Yung. Non-interactive cryptocomputing for NC^1. In *Proceedings of 40th Annual IEEE Symposium on Foundations of Computer Science*, pages 554–566, 1999.

[39] D. Beaver. Minimal-latency secure function evaluation. *Advances of Cryptology(EUROCRYPT 2000)*, LNCS 1807:335–350, 2000.

[40] M. Naor and K. Nissim. Communication preserving protocols for secure function evaluation. In *Proceedings of the Thirty-third Annual ACM Symposium on Theory of Computing*, pages 590–599. ACM Press, 2001.

[41] R. Canetti. Security and composition of multiparty cryptographic protocols. *Journal of Cryptology*, 13(1):143–202, 2000.

[42] M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *Proceedings of the First ACM Conference on Computer and Communications Security*, pages 62–73. ACM Press, 1993.

[43] R. Canetti, O. Goldreich, and S. Halevi. The random oracle methodology, revisited (preliminary version). In *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*, pages 209–218. ACM Press, 1998.

[44] M. Rabin. How to exchange secrets by oblivious transfer. In *Technical Report, TR-81, Aiken Computation Laboratory*, 1981.

[45] J. Halpern and M. Rabin. A logic to reason about likelihood. In *Proceedings of the Fifteenth Annual ACM Symposium on Theory of Computing*, pages 310–319. ACM Press, 1983.

[46] S. Even, O. Goldreich, and A. Lempel. A randomized protocol for signing contracts. *Communications of the ACM*, 28(6):637–647, 1985.

[47] G. Brassard, C. Crépeau, and J.M. Robert. All-or-nothing disclosure of secrets. *Advances in Cryptology (CRYPTO 1986)*, LNCS 263:234–238, 1987.

[48] G. Brassard, C. Crépeau, and J.M. Robert. Information theoretic reductions among disclosure problems. In *Proceedings of Twenty-seventh Annual IEEE Symposium on Foundations of Computer Science*, pages 168–173. IEEE, 1986.

[49] C. Crépeau. Equivalence between two flavours of oblivious transfers. *Advances in Cryptology (CRYPT0 1987)*, LNCS 293:350–354, 1988.

[50] R. Impagliazzo and S. Rudich. Limits on the provable consequences of one-way permutations. In *Proceedings of the Twenty-first Annual ACM Symposium on Theory of Computing*, pages 44–61. ACM Press, 1989.

[51] M. Bellare and S. Micali. Non-interactive oblivious transfer and applications. *Advances in Cryptology (CRYPTO 1989)*, LNCS 435:547–557, 1990.

[52] M. Naor and B. Pinkas. Oblivious transfer and polynomial evaluation. In *Proceedings of the Thirty-first Annual ACM Symposium on Theory of Computing*, pages 245–254. ACM Press, 1999.

[53] M. Naor and B. Pinkas. Efficient oblivious transfer protocols. In *Proceedings of the Twelfth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 448–457. Society for Industrial and Applied Mathematics, 2001.

[54] B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan. Private information retrieval. In *Proceedings of the 36th Annual IEEE Symposium on Foundations of Computer Science*, pages 41–50. IEEE, 1995.

[55] B. Chor and N. Gilboa. Computationally private information retrieval (extended abstract). In *Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing*, pages 304–313. ACM Press, 1997.

[56] E. Kushilevitz and R. Ostrovsky. Replication is not needed: single database, computationally-private information retrieval. In *Proceedings of the 38th Annual Symposium on Foundations of Computer Science*, pages 364–373. IEEE Computer Society, 1997.

[57] C. Cachin, S. Micali, and M. Stadler. Computationally private information retrieval with polylogarithmic communication. *Advances in Cryptology (EUROCRYPT 1999)*, LNCS 1592:402–414, 1999.

[58] Y. Gertner, Y.Ishai, E.Kushilevitz, and T.Malkin. Protecting data privacy in private information retrieval schemes. In *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*, pages 151–160. ACM Press, 1998.

[59] B. Huberman, M. Franklin, and T.Hogg. Enhancing privacy and trust in electronic communities. In *Proceedings of the First ACM Conference on Electronic Commerce*, pages 78–86. ACM Press, 1999.

[60] J. Vaidya and C. Clifton. Secure set intersection cardinality with application to association rule mining. Accepted for Publication in the Journal of Computer Security, IOS Press.

[61] J. Vaidya and C. Clifton. Leveraging the "multi" in secure multi-party computation. In *Proceedings of the ACM Workshop on Privacy in the Electronic Society*, pages 53–59. ACM Press, 2003.

[62] M. Freedman, K. Nissim, and B. Pinkas. Efficient private matching and set intersection. *Advances in Cryptology (EUROCRYPT 2004)*, 2004.

[63] G. Aggarwal, N. Mishra, and B. Pinkas. Secure computation of the kth-ranked element. In *Advances in Cryptology (EUROCRYPT 2004)*, 2004.

[64] E. Kushilevitz and N. Nisan. *Communication Complexity.* Cambridge University Press, 1997.

[65] C. Cachin. Efficient private bidding and auctions with an oblivious third party. In *Proceedings of the Sixth ACM Conference on Computer and Communications Security*, pages 120–127. ACM Press, 1999.

[66] M. Fischlin. A cost-effective pay-per-multiplication comparison method for millionaires. In *CT-RSA*, pages 457–472, 2001.

[67] M. Franklin and M. Reiter. The Design and Implementation of a Secure Auction Service. In *Proc. IEEE Symposium on Security and Privacy*, pages 2–14, Oakland, Ca, 1995. IEEE Computer Society Press.

[68] M. Harkavy, J.D. Tygar, and H. Kikuchi. Electronic auctions with private bids. In *3rd USENIX Workshop on Electronic Commerce*, pages 61–74, September 1998.

[69] H. Kikuchi. (m+1)st-price auction protocol. In *Financial Cryptography – Fifth International Conference*, LNCS 2339, pages 351–363, 2001.

[70] F. Brandt. Fully private auctions in a constant number of rounds. In *Financial Cryptography – Seventh International Conference*, to appear, 2003.

[71] F. Brandt. A verifiable, bidder-resolved auction protocol. In *Fifth International Workshop on Deception, Fraud and Trust in Agent Socities*, pages 18–25, 2002.

[72] F. Brandt and T. Sandholm. Efficient privacy-preserving protocols for multi-unit auctions. In *Financial Cryptography – Ninth International Conference*, pages 296–310, 2005.

[73] O. Baudron and J. Stern. Non-interactive private auctions. In *Financial Cryptography – Fifth International Conference*, LNCS 2339, pages 364–378, 2001.

[74] H. Lipmaa, N. Asokan, and V. Niemi. Secure vickrey auctions without threshold trust. In *Financial Cryptography – Sixth International Conference*, 2003.

[75] M. Naor, B. Pinkas, and R. Sumner. Privacy preserving auctions and mechanism design. In *Proceedings of the First ACM Conference on Electronic Commerce*, pages 129–139. ACM Press, 1999.

[76] I. Damgård and M. Jurik. A generalisation, a simplification and some applications of paillier's probabilistic public-key system. In *4th International Workshop on Practice and Theory in Public Key Cryptosystems, PKC 2001*, LNCS 1992, pages 119–136, 2001.

[77] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. *Advances in Cryptology (EUROCRYPT 1999)*, LNCS 1592:223–238, 1999.

[78] T. Okamoto, S. Uchiyama, and E. Fujisaki. Epoc: Efficient probabilistic public-key encryption, 1998.

[79] A. Shamir. Identity-based cryptosystems and signature schemes. *Advances in Cryptology (CRYPTO 1984)*, LNCS 196:47–53, 1984.

[80] D. Boneh and M. Franklin. Identity-Based Encryption from the Weil Pairing. *Advances in Cryptology (CRYPTO 2001)*, LNCS 2139:213–229, 2001.

[81] C. Cocks. An identity based encryption scheme based on quadratic residues. In *8th IMA International Conference on Cryptography and Coding*, volume 2260, pages 360–363. Springer, December 2001.

[82] W. Du. *A Study of Several Specific Secure Two-party Computation Problems.* PhD thesis, Purdue University, West Lafayette, Indiana, USA, 2001.

[83] W. Du and M. J. Atallah. Privacy-preserving statistical analysis. In *Proceedings of the Seventeenth Annual Computer Security Applications Conference*, pages 102–110, December 10-14 2001.

[84] J. Vaidya and C. Clifton. Privacy preserving association rule mining in vertically partitioned data. In *KDD '02: Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 639–644. ACM Press, 2002.

[85] B. Goethals, S. Laur, H. Lipmaa, and T. Mielikainen. On private scalar product computation for privacy-prerving data mining. In *The Seventh Annual International Conference on Information Security and Cryptology (ICISC 2004)*, 2004.

[86] J. Holt, R. Bradshaw, K. Seamons, and H. Orman. Hidden credentials. In *Proceedings of the Second ACM Workshop on Privacy in the Electronic Society*, October 2003.

[87] P. Bonatti and P. Samarati. Regulating service access and information release on the web. In *Proceedings of the Seventh ACM Conference on Computer and Communications Security*, pages 134–143. ACM Press, November 2000.

[88] K. Seamons, M. Winslett, and T. Yu. Limiting the disclosure of access control policies during automated trust negotiation. In *Proceedings of the Symposium on Network and Distributed System Security*, February 2001.

[89] K. Seamons, M. Winslett, T. Yu, L. Yu, and R. Jarvis. Protecting privacy during on-line trust negotiation. In *Second Workshop on Privacy Enhancing Technologies*. Springer-Verlag, April 2002.

[90] W. Winsborough and N. Li. Protecting sensitive attributes in automated trust negotiation. In *Proceedings of the ACM Workshop on Privacy in the Electronic Society*, pages 41–51. ACM Press, November 2002.

[91] W. Winsborough and N. Li. Towards practical automated trust negotiation. In *Proceedings of the Third International Workshop on Policies for Distributed Systems and Networks*, pages 92–103. IEEE Computer Society Press, June 2002.

[92] W. Winsborough and N. Li. Safety in automated trust negotiation. In *Proceedings of IEEE Symposium on Security and Privacy.* IEEE Computer Society Press, May 2004.

[93] W. Winsborough, K. Seamons, and V. Jones. Automated trust negotiation. In *DARPA Information Survivability Conference and Exposition*, volume I, pages 88–102. IEEE Press, January 2000.

[94] M. Winslett, T. Yu, K. Seamons, A. Hess, Jared Jacobson, Ryan Jarvis, Bryan Smith, and Lina Yu. Negotiating trust on the web. *IEEE Internet Computing*, 6(6):30–37, November/December 2002.

[95] T. Yu, X. Ma, and M. Winslett. Prunes: An efficient and complete strategy for trust negotiation over the internet. In *Proceedings of the Seventh ACM Conference on Computer and Communications Security*, pages 210–219. ACM Press, November 2000.

[96] T. Yu and M. Winslett. A unified scheme for resource protection in automated trust negotiation. In *Proceedings of IEEE Symposium on Security and Privacy*, pages 110–122. IEEE Computer Society Press, May 2003.

[97] T. Yu, M. Winslett, and K. Seamons. Interoperable strategies in automated trust negotiation. In *Proceedings of the Eighth ACM Conference on Computer and Communications Security*, pages 146–155. ACM Press, November 2001.

[98] N. Li, W. Du, and D. Boneh. Oblivious signature-based envelope. In *Proceedings of the 22nd ACM Symposium on Principles of Distributed Computing.* ACM Press, July 2003.

[99] R. Bradshaw, J. Holt, and K. Seamons. Concealing complex policies with hidden credentials. In *Proceedings of Eleventh ACM Conference on Computer and Communications Security*, October 2004.

[100] Equifax. URL `http://www.equifax.com`.

[101] Experian. URL `http://www.experian.com`.

[102] Transunion. URL `http://www.transunion.com`.

[103] L. Valiant. Universal circuits (preliminary report). In *Proceedings of the Eighth Annual ACM Symposium on Theory of Computing*, pages 196–203, New York, NY, USA, 1976. ACM Press.

[104] E. Kushilevtiz. Privacy and communication complexity. In *Proceedings of the Thirtieth Annual IEEE Symposium on Foundations of Computer Science*, pages 416–421, 1989.

[105] M. Atallah and W. Du. Secure multi-party computational geometry. *Lecture Notes in Computer Science*, 2125:165–179, 2000.

[106] D. Robbins. Average distance between two points in a box. *American Mathematical Monthly*, 85, 278, 1978.

[107] D. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24(2):84–90, 1981.

[108] B. Grosof, Y. Labrou, and H. Chan. A declarative approach to business rules in contracts: courteous logic programs in xml. In *Proceedings of the First ACM Conference on Electronic Commerce*, pages 68–77, New York, NY, USA, 1999. ACM Press.

[109] G. Governatori, A. ter Hofstede, and P. Oaks. Defeasible logic for automated negotiation. In P. Swatman and P.M. Swatman, editors, *Proceedings of CollECTeR*. Deakin University, 2000. Published on CD.

[110] R. Smith and J. Shao. Preserving privacy when preference searching in e-commerce. In *Proceeding of the ACM workshop on Privacy in the Electronic Society*, pages 101–110. ACM Press, 2003.

[111] G. Davida, Y. Frankel, and B. Matt. On enabling secure applications through off-line biometric identification. In *Proceedings of 1998 IEEE Symposium on Security and Privacy*, pages 148–157, May 1998.

[112] G. Bleumer. Offine personal credentials. Technical Report TR 98.4.1, AT&T, 1998.

[113] G. Bleumer. Biometric yet privacy protecting person authentication. In *Proceedings of 1998 Information Hiding Workshop*, pages 101–112. Springer-Verlag, 1998.

[114] R. Impagliazzo and S. More. Anonymous credentials with biometrically-enforced non-transferability. In *Proceedings of the Second ACM Workshop on Privacy in the Electronic Society (WPES '03)*, pages 60–71, October 2003.

[115] D. Chaum and T. Pedersen. Wallet databases with observers. *Advances in Cryptology (Crypto 1992)*, 740:89–105, 1993.

[116] G. Davida, Y. Frankel, and B. Matt. On the relation of error correction and cryptography to an off-line biometric based identification scheme. In *Proceedings of WCC99, Workshop on Coding and Cryptography*, 1999.

[117] A. Juels and M. Wattenberg. A fuzzy commitment scheme. In *Proceedings of the Sixth ACM Conference on Computer and Communications Security*, pages 28–36. ACM Press, 1999.

[118] G. Davida and Frankel Y. Perfectly secure authorization and passive identification for an error tolerant biometric system. In *Proceedings of Seventh Conference on Cryptography and Coding, LNCS 1746*, pages 104–113, 1999.

[119] A. Juels and M. Sudan. A fuzzy vault scheme. In *Proceedings of the 2002 IEEE International Symposium on Information Theory*, pages 408–413, 2002.

[120] T. Clancy, N. Kiyavashr, and D. Lin. Secure smartcard-based fingerprint authentication. In *Proceedings of the 2003 ACM Workshop on Biometrics Methods and Applications*, pages 45–52, 2003.

[121] Y. Dodis, L. Reyzin, and A. Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. *Advances in Cryptology (EUROCRYPT 2004)*, pages 523–540, 2004.

[122] F. Kerschbaum, M. Atallah, D. Mraihi, and J. Rice. Private fingerprint verification without local storage. In *International Conference on Biometric Authentication (ICBA)*, July 2004.

[123] G. Hachez, F. Koeune, and J. Quisquater. Biometrics, access control, smart cards: A not so simple combination. In *Proceedings of the Fourth Working Conference on Smart Card Research and Advanced Applications*, pages 273–288. Kluwer Academic Publishers, September 2000.

[124] R. Bolle, J. Connell, and N. Ratha. Biometric perils and patches. *Pattern Recognition*, 35(12):2727–2738, 2002.

[125] B. Schneier. Biometrics: Truths and fictions. URL `ttp://www.scneier.com/crypto-gram-9808.html#biometrics`.

[126] R. Anderson and M. Kuhn. Tamper resistance - A cautionary note. In *Proceedings of the 2nd USENIX Workshop on Electronic Commerce*, pages 1–11, 1996.

[127] R. Anderson and M. Kuhn. Low cost attacks on tamper resistant devices. In *International Workshop on Security Protocols*, pages 125–136, 1997.

[128] A.K. Jain, L. Hong, and R. Bolle. On-line fingerprint verification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(4):302–314, 1997.

VITA

VITA

Keith Frikken was born in 1979. In May 2000, he graduated from Winona State University with a bachelor's degree in computer science and mathematics. He spent the following summer as an intern at IBM. In August 2000, he started his graduate education at Purdue University. In August 2005, he received the degree of Doctor of Philosophy under the supervision of Mikhail Atallah. His research interests include information security and databases.