Flexible, Secure and Private Point-based Trust Management

Danfeng Yao, Keith B. Frikken,
Mikhail J. Atallah, Roberto Tamassia

Department of Computer Science
Brown University
Providence, Rhode Island 02912

CS-06-07
April 2006

Flexible, Secure and Private Point-based Trust Management

November 4, 2005

Abstract

There has been much recent work on privacy-preserving access control negotiations, i.e., carrying out the negotiation in a manner that minimizes the disclosure of credentials and of access policies. This paper introduces the notion of point-based policies for access control and gives protocols for implementing them in a disclosure-minimizing fashion. Specifically, Bob values each credential with a certain number of points and requires a minimum total threshold of points before granting access to a resource to Alice. In turn, Alice values each of her credentials with a privacy score that indicates her reluctance to reveal that credential. She is interested in achieving the required threshold for accessing the resource while minimizing the sum of the privacy scores of her used credentials. Bob's valuation of credentials is private and should not be revealed, as is his threshold. Alice's privacy-valuation of her credentials is also private and should not be revealed. What Alice uses is a subset of her credentials that achieves Bob's required threshold for access, yet is of as small a value to her as possible.

We give a protocol for computing such a subset of Alice's credentials without revealing any of the two parties' above-mentioned sensitive valuation functions and threshold numbers. The main advantages of our scheme are that (i) it prevents pre-mature information leakage when the trust establishment is unsuccessful; (ii) it efficiently supports cumulative privacy, which we define as the sensitivity of a combination of credentials being additive; and (iii) it mitigates the danger to both parties from probing attacks by their counterpart. Note that, in the proposed point-based scheme, two individuals with identical credentials have the flexibility of satisfying the access threshold in very different ways, depending on their own individual privacy criteria.

A contribution of this paper that goes beyond the specific problem considered is a general method for recovering an optimal solution from any value-computing dynamic programming computation, while detecting cheating by the participants. Specifically, our traceback technique relies on the subset sum problem to force consistency.

1 Introduction

Access decisions in stand-alone systems are usually based on the identity of the entity requesting a resource, whereas in open systems such as the Internet, this approach becomes ineffective. This is because the resource owner and the client (or requester) typically belong to different security domains controlled by different authorities and are unknown to each other. The modern alternative is to use *digital credentials* for satisfying access policies. Digital credentials are digitally signed assertions about the credential owner by a credential issuer. Each digital credential contains a set of attributes about the owner. The decision to access a resource is based on the attributes in the client'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 client, 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. However, this mechanism is unacceptable if the credentials or the access control policies are considered to be sensitive information. The motivation for hiding credentials is

individual privacy, e.g., if the credentials are about one's physical impairment or disability, financial distress, political or religious affiliation, etc. The motivation for hiding the policy is not only 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 usually renders an 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.

It is also important to point out that a process that treats Alice's credentials as confidential is ultimately not only to Alice's advantage but also to Bob's: Bob can worry less 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.

This paper is the first access control work that quantitatively addresses degrees of sensitivity, instead of the traditional coarse labels of sensitive/non-sensitive. Moreover, the degree of sensitivity of a given credential is private to each user, and can vary from one user to another. Our system is not a mere fine-grain privacy-preserving generalization of Multi-Level Security (MLS) [45] because of its additive nature, e.g., in an MLS-like scheme it is not possible to aggregate a number of lower-level clearances to access higher-clearance documents. However, our system subsumes MLS in that it can simulate it through a judicious assignment of quantitative values (e.g., by assigning large enough values to higher levels that they are unreachable through any additive aggregation of lower credentials).

1.1 Motivations

In a probing attack, Alice can engage in a protocol with Bob multiple times using different credential sets each time (all of which are subsets of her credentials) to gain information about Bob's policy. In the case where Alice is requesting access to a service, Bob will know whether she got access and can therefore also probe (by using different policies and observing their effect) to gain information about Alice's credentials.

One way of mitigating probing attacks is the one followed in the trust negotiation literature [46, 49, 50, 51, 54, 55, 56], in which the disclosure of a credential is governed by an access control policy that specifies the prerequisite conditions that must be satisfied in order for that credential to be disclosed. Typically, the prerequisite conditions are a subset of the set of all credentials, and the policies are modeled using propositional formulas. A trust negotiation protocol is normally initiated by a client requesting a service or a resource from a server, and the negotiation consists of a sequence of credential exchanges: Trust is established if the initially requested service or resource is granted and all policies for disclosed credentials are satisfied [51, 54]. The sequence of credentials disclosed during this negotiation depends on which strategy each negotiator uses for controlling which credentials are disclosed and when to disclose them, and when to terminate a negotiation [56]. Several negotiation strategies are proposed in [51, 54, 56]. For example, in the eager strategy [51], two negotiators take turns disclosing a credential to the other side as soon as the access control policy for that credential is satisfied.

Although mitigating probing attacks, the requirements of the trust negotiation literature represent important practical constraints. **(1)** Probing is still possible when policies are not treated as sensitive resources, and the client (or server) can game the system in many ways. For example, if the client knows the access control policies for the server's credentials then she will know the path of least resistance to unlock certain credentials. **(2)** Pre-mature information leaking is difficult to prevent in existing trust negotiation protocols including the framework using cryptographic credentials by Li, Li, and Winsborough [39]. The pre-mature information leaking refers to the situation when a negotiation is not successful, however sensitive credentials of a negotiator are already disclosed. **(3)** The service model in trust negotiation is usually limited, that is,

the requested service is fixed and independent of the amount of information released by the client at the end of the negotiation session. However, a client may end up disclosing more information than what is required for the initially requested service. The reward or service provided by the server should be dynamically adjustable with the valuable information released from the client.

In addition, an important notion of *cumulative privacy* cannot be efficiently supported by trust negotiation protocols. In existing solutions, revealing a credential has little to do with the credentials that have already been revealed. For example, Alice may consider her birthday or her birth-town to be insensitive, but consider the combination of them to be sensitive, as it would isolate her into a small group of people. Cumulative privacy is a desirable notion of privacy. It represents the cumulative sensitivity of a combination of credentials, which may or may not be sensitive themselves. Supporting cumulative privacy in trust negotiation is not efficient, as it requires the specifications of an exponential number of policies. As will become clear soon, the model and protocols presented in this paper provide an alternative solution for trust establishment that improves on the security and flexibility of existing approaches.

1.2 Overview

We consider a point-based trust management policy (rather than a Boolean expression) that is private and should therefore not be revealed to Alice: Bob associates a number of points with every possible credential, and requires the sum of the points of those credentials that Alice uses to reach a minimum threshold before he grants her access to the resource (each resource can have its own threshold, and that threshold is itself private and should not be revealed to Alice). Alice needs to satisfy the threshold requirement to gain access by using a subset of her credentials that gives her the required number of points, but there can be many such subsets: Alice is interested in using the subset that has minimum privacy-value to her, according to her privacy-valuation function; that valuation function is itself private and should not be revealed to Bob. We give a protocol which determines which subset of Alice's credentials optimally satisfies Bob's threshold, i.e., it has minimum privacy value to Alice among all subsets that satisfy Bob's threshold (Bob's point-valuation of credentials, his thresholds, and Alice's privacy-valuation of her credentials, are all private and not revealed by the protocol). What happens after that subset is computed is flexible: If the privacy score of that subset is low enough then Alice may choose to simply reveal those credentials to Bob, but if the score is high then she may insist on engaging in another protocol (such as [34, 11, 29, 27]) in which she proves to Bob that she has such a subset that satisfies his points-threshold requirement but without revealing to him the elements of that subset (this will require the usual off-line third-party certification of the credentials that is used in the previous protocols).

One of our main contributions is the formulation of the point-based trust management model. This model is the first scheme that prevents pre-mature information leakage. It gives the privacy protection to the client even when the trust establishment is unsuccessful, which may happen if the client thinks the total privacy scores of to-be disclosed credentials is too high.

Our model facilitates the notion cumulative privacy, that is revealing multiple credentials may be more costly from a privacy standpoint than revealing a single credential. Our point-based model intrinsically captures the additive characteristic of user privacy; the privacy of credentials is relative to what else have been released. This is an accurate model for privacy protection that reflects an intuitive notion of privacy. Our specification of privacy scores has size linear in the number of credentials, which is more efficient than what would be required for trust negotiation.

The point-based model explicitly associates credentials with values obtained from service provider, therefore the client's reward or service can be dynamically adjusted according to the amount of information released. This flexibility makes the point-based model attractive to the trust management in web-services and e-commerce applications in general, as users have the incentives to carry on the computation for trust establishment, which facilitates business transactions.

1.3 Summary of contributions

We develop the first point-based trust management model that can individually and quantitatively address consumers' privacy concerns in e-commerce transactions. The model enables users to quantitatively distinguish the sensitivities of different credentials. It also allows a provider to quantitatively assign values to a user's credential. The point-based trust management facilitates the trust establishment between the two parties, and empowers the users to pro-actively and quantitatively control the release of their sensitive information. We summarize our contributions as follows.

1. We formalize the credential selection problem in our point-based trust management model into a knapsack problem, and give secure and private dynamic programming protocols for solving the knapsack problem. Our solution, consisting of a basic protocol and an improved protocol, allows the server and user to jointly compute the optimal sum of privacy scores for the released credentials, without revealing their private parameters.
   The communication complexity of our basic protocol for dynamic programming is $O(nT')$, where n is the total number of credentials and T' is the *marginal threshold*. Marginal threshold is defined by us, and corresponds to the sum of the points of the credentials that are *not* disclosed. At each round of the communication, each party performs $O(1)$ homomorphic encryption operations. The basic protocol is semantic-secure under a semi-honest adversarial model.
   Our improved protocol, *Extended Protocol II*, is resilient against stronger adversaries than semi-honest ones (a.k.a honest-but-curious), in that it prevents a participant from tampering with the values used in the dynamic programming computation. That is, while we cannot prevent a participant from lying about her input, we can force *consistency in lying* by preventing capricious use of different inputs during the crucial solution-traceback phase. For complex functions such as the one being studied, lying about one's input wrecks the worthiness of the answer for both participants, and the participant who does so would have been better off not engaging in the protocol in the first place (this is not true for simple functions where the lier can still get the answer by *correcting for her lie*). For the extended protocol II, the computation cost for both the server and the client, and the communication complexity are $O(n^2T')$.
2. One contribution of this paper that goes beyond the specific problem considered is a general *indexing expansion* method for recovering an optimal solution from any value-computing dynamic programming computation, while detecting cheating by the participants. Specifically, our traceback technique relies on the subset sum problem and random information checksum to enforce the consistency of a participant. Using this method, a participant is not required to trust the other party during the backtracing phase. This is possible because the participant is able to efficiently identify whether the other party has tampered with the computation. Furthermore, sensitive parameters used by both parties remain private in the protocol. For the traceback in general dynamic programming problems, our algorithm not only allows a participant to independently and easily recover the optimal traceback solution, once the computed optimal value is given, but also enables the participants to verify the integrity of the optimal value.

1.4 Organization of the paper

The paper is organized as follows. Related work is given in Section 2. Our point-based trust management model is presented in Section 3. The basic protocol for computing the dynamic programming table is given in Section 4. Our basic and improved traceback algorithms for identifying the selected credentials from the dynamic programming table are presented in Section 5 and 6, respectively. We analyze the security of our protocols in Section 7. Conclusions and future work are given in Section 8. In the appendices, our security model and proofs are given.

2 Related Work

In this section, we discuss the existing work on secure multi-party computation, access control including trust negotiation and hidden credentials, and on economics of information security. The protocols in this paper are compared with the existing work.

2.1 Access control and trust management systems

There has been extensive research on access control models in the past decade [25, 43, 44]. Trust management models are developed for the authorization in distributed systems [6, 7, 9, 15, 19, 40, 48]. The concept of role-based access control [25, 44] is widely deployed to improve the scalability and efficiency of management.

Our point-based trust management model quantitatively treats memberships or credentials, which is significantly different from the existing access control models. Our approach aims to address the fact that different individuals or groups of people have different privacy concerns in terms of protecting sensitive information. This goal differs significantly from the somewhat rigid conventional access control models. The flexibility provided by the point-based model enables users to pro-actively protect their private information. Although flexible, our access control model still offers strong protection for the resources. Thresholds specified by resource owners prevent unqualified users from accessing the resource.

Holt et al. proposed a novel hidden credentials system [34] that protects sensitive credentials and policies. The main idea of that paper is that when a signature derived from an identity based encryption scheme (IBE) [10, 20, 47] is used to sign a credential, the credential content can be used as a public encryption key such that the signature is the corresponding decryption key. Hidden credentials can be used in such a way that they are never shown to anyone, thus the sensitive credentials are protected. Bradshaw et al. [11] 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. Frikken et al. [29] give a scheme that hides both credentials and policies. Most recently, Frikken et al. [27] proposed a protocol that allows the client and the servers to have policies for their credentials (to mitigate probing attacks) and hide these policies and the credentials. The above-mentioned work is in the conventional access control settings, where the server and authorized clients have established trust when hidden credentials are issued. Our point-based trust management model is suitable for more general contexts where the server and clients are initially unknown to each other.

Anonymous credential systems have been developed [12, 16, 18] to allow anonymous yet authenticated and accountable transactions between users and service providers. The systems give a technique for protecting users' privacy when conducting Internet transactions. Our work proposed in this paper is orthogonal to anonymous credential systems. We can apply anonymous credential techniques to implement membership credentials in our point-based trust management model. These credentials are then used to prove user's memberships without revealing individual identity.

We have compared the trust negotiation protocols [46, 49, 50, 51, 54, 55, 56] with our point-based trust management model in the introduction. Recently, Li, Li, and Winsborough introduce a framework for trust negotiation in which the diverse credential schemes and protocols including anonymous credential systems can be combined, integrated, and used as needed [39]. The paper presents a policy language that enables negotiators to specify authorization requirements. Our work differs from the existing trust negotiation frameworks in that (i) we provide a quantitative approach to the credential-disclosure problem, which allows flexible selections of released information (ii) the ability to assign a privacy score to each individual credential allows fine-grained privacy control for users, which can be more flexible and efficient than using a combination of different trust negotiation strategies (iii) probing attacks are largely mitigated in our model and protocols, especially when the transaction is not established.

2.2 Secure multi-party computation

Secure Multi-party Computation (SMC) was introduced in a seminal paper by Yao [52], which contained a scheme for secure comparison. Suppose Alice (with input a) and Bob (with input b) desire to determine whether or not $a < b$ without revealing any information other than this result (this is known as *Yao's Millionaire Problem*). More generally, SMC allows Alice and Bob with respective private inputs a and b to compute a function $f(a, b)$ by engaging in a secure protocol for public function f. Furthermore, the protocol is private in that it reveals no additional information. This means that Alice (Bob) learns nothing other than what can be deduced from a (b) and $f(a, b)$. Elegant general schemes are given in [5, 17, 30, 32] for computing any function f privately.

Besides the generic work in the area of SMC, there has been extensive work on the privacy-preserving computation of various functions. For example, computational geometry [2, 24], privacy-preserving computational biology [1]. The private dynamic programming protocol given by Atallah and Li [1] is the most relevant work to ours. Their protocol compares biological sequences in an additively split format. Each party maintains a matrix, and the summation of two matrices is the real matrix implicitly used to compute the edit distance. Our protocols also carry out computation in an additively split form. What distinguishes us from existing solutions is that we are able to achieve efficiently a stronger security guarantee. We construct a protocol that is secure against a stronger adversary than a semi-honest one without using Zero-Knowledge Proofs [33]. Recently, there are also solutions for privacy-preserving automated troubleshooting [35], privacy-preserving distributed data mining [36], private set operations [26, 37], and equality tests [41]. We do not enumerate other private multi-party computation work as their approaches significantly different from ours.

2.3 Pricing security and privacy

A novel category of interdisciplinary work on the economics of information security and privacy has recently been proposed [13, 23, 38]. Economics and psychology knowledge is applied to the problem of pricing the worth of security externalities and privacy information. In a recent study, Danezis, Lewis, and Anderson obtain an estimate of the value that users attach to their location data [23]. These important studies and analysis lay the foundation for applications that deploy our point-based model.

3 Model

In this section, we describe a point-based trust management model, and define the credential selection problem in this model.

3.1 Point-Based Trust Management

In the point-based trust management model, the authorization policies of a resource owner defines an *access threshold* for each of its resources. The threshold is the minimum amount of points required for a requester to access that resource. For example, accessing a medical database requires fifty points. The resource owner also defines a *point value* for each type of credentials, which denotes the number of points or credits a requester obtains if a type of credential is disclosed. For example, a valid ACM membership has ten points. This means that a user can disclose his or her ACM membership credential in exchange for ten points. We call this a trust management model as opposed to an access control model, because the resource owner does not know the identities or role assignments of requesters *a priori* as in conventional access control settings.

A requester has a set of credentials, and some of which may be considered sensitive and cannot be disclosed to everyone. However, in order to access a certain resource, the requester has to disclose a number of credentials such that the access threshold for that resource is met by the points of the disclosed credentials. Different users may have different perspective on the sensitivity of their credentials, even though the credentials are of the same type. For example, a teenager may consider the age information insensitive, whereas a middle-aged person may not be very willing to tell his or her age.

Therefore, in point-based trust management model, each user defines a *privacy score* for each of their credentials. The privacy score represents the inverse of the willingness to disclose a credential. For example, Alice may give a privacy score of ten to her college ID, and give fifty to her credit card. The user is granted access to a certain resource if the access point threshold is met and all of the disclosed credentials are valid. Otherwise, the access is denied. From the requester's point of view, one central question is how to fulfill the access threshold while disclosing the least amount of sensitive information. We now define this as a credential selection problem. Solving the credential selection problem is challenging, because the requester considers his or her privacy scores sensitive, and the server considers its point values and access threshold sensitive.

3.2 Credential Selection Problem

Definition 1 *The credential selection problem is to determine an optimal combination of requester's credentials to disclose to the resource owner, such that the minimal amount of sensitive information is disclosed and the access threshold of the requested resource is satisfied by the disclosed credentials.*

We formalize the credential selection problem as an optimization problem. Our model assumes that the resource owner (or server) and the requester (or user) agree on a set of credential types as the universe of credentials $(C_1, \ldots, C_n)$. We define a binary vector $(x_1, \ldots, x_n)$ as the unknown variable to be computed, where x_i is 1 if credential C_i is selected and 0 if otherwise. Integer variable $a_i \geq 0$ is the *privacy score* of credential C_i. It is assigned by the requester *a priori*. If the requester does not have a certain credential C_i, the privacy score a_i for that credential can be set to any integer larger than T, where T is the trust threshold for the requested resource. Integer variable $p_i \geq 0$ is the point value for releasing credential type C_i. The requester considers all a_i values sensitive, and the server considers all p_i values sensitive.

The credential selection problem is for the requester to compute a binary vector $(x_1, \ldots, x_n)$ such that the sum of points $\sum_{i=1}^{n} x_i p_i$ satisfies T, and the sum of privacy scores $\sum_{i=1}^{n} x_i a_i$ is minimized. This is captured in the following minimization problem. Compute a binary vector $(x_1, \ldots, x_n)$ such that the following holds:

$$\begin{aligned} \min \quad & \sum_{i=1}^{n} x_i a_i \\ \text{subject to} \quad & \sum_{i=1}^{n} x_i p_i \geq T \end{aligned}$$

The above minimization problem can be rewritten into a knapsack problem with new variable $y_i = 1 - x_i \in \{0, 1\}$. For i-th credential, $y_i = 1$ represents not disclosing the credential, and $y_i = 0$ represents disclosing the credential.

$$\begin{aligned} \max \quad & \sum_{i=1}^{n} y_i a_i \\ \text{subject to} \quad & \sum_{i=1}^{n} y_i p_i < \sum_{i=1}^{n} p_i - T \end{aligned}$$

The dynamic programming solution for the knapsack problem is pseudo-polynomial: the running time is in $O(nT')$, where $T' = \sum_{i=1}^{n} p_i - T$. We refer to T' as the *marginal threshold*. Marginal threshold coarsely correlates to the sum of points of credentials that are not disclosed.

Definition 2 *The marginal threshold T' of the credential selection problem is defined as $\sum_{i=1}^{n} p_i - T$, where p_i is the point value for credential type C_i, T is the access threshold for a requested resource, and n is the total number of credential types.*

Let us first review the dynamic programming solution for the 0/1 knapsack problem [21]. Then, we describe our protocol for carrying out private dynamic programming computation of the knapsack problem. The 0/1 knapsack problem is defined as follows. Given items of different integer values and weights, find the most valuable set of items that fit in a knapsack of fixed integer capacity. In the dynamic programming of knapsack problem, a table is made to track the optimal selection of items so far. A column indicates the range of values, which corresponds to the target weight of the knapsack. A row corresponds to each item. The table stops at the maximum capacity of the knapsack. The first column and the first row are initialized to zeros, i.e. $M_{0,j}$ and $M_{i,0}$ are zeros, for all $i \in [1, n]$ and $j \in [0, T']$. The table is filled from top to bottom and from left to right. Using the notations defined earlier, the recurrence relation is formally defined as follows.

Denote $M_{i,j}$ as the value at i-th row and j-th column, and $i \in [0, n], j \in [0, T']$.

$$M_{i,j} = \begin{cases} M_{i-1,j} & \text{if } j < p_i \\ \max\{M_{i-1,j}, M_{i-1,j-p_i} + a_i\} & \text{if } j \geq p_i \end{cases}$$

Each entry of the table stores the total value of a knapsack, which is determined as either the value of a knapsack without the current item (expressed as the value directly to the top of the current entry), or the value of the knapsack with the current item added into it. At the end of the computation, the entry at the lower right corner of the table contains the optimal value of the knapsack. The selections of items can be obtained by book-keeping the information of where the value of an entry comes from.

For our credential selection problem, the above recurrence relation means the following. If the point value of credential type C_i exceeds j, which is a value in the range of $[0, T']$, then the i-th credential is not selected and the privacy score $M_{i,j}$ is kept the same as $M_{i-1,j}$. Otherwise, compare the score $M_{i-1,j}$ associated with not selecting the i-th credential and the score $M_{i-1,j-p_i} + a_i$ for selecting the i-th credential. The larger value is chosen to be the privacy score $M_{i,j}$.

The standard dynamic programming computation requires values a_i and p_i for all $i \in [1, n]$. However, in our model, the requester considers a_i sensitive, and the server considers p_i sensitive. We present a protocol that allows the completion of the dynamic programming computation without revealing any sensitive information. In addition to protecting sensitive a_i and p_i values, the entries in the dynamic programming table are also protected from both parties. From this perspective, our protocol provides better privacy protection than the secure multi-agent dynamic programming work by Yokoo and Suzuki [53], as their approach cannot prevent the disclosure of entries in the table.

4 Point-Based Trust Management: Basic Protocol

We present the *basic protocol*, which is a secure two-party protocol for computing the dynamic programming table (matrix) in our credential selection problem. The protocol maintains the secrecy of sensitive parameters of both parties. Furthermore, neither the server nor the user learns any intermediate result. The main technical challenge is that the server does not want to reveal point values $\{p_i\}$ and the client does not want to reveal privacy scores $\{a_i\}$. As shown by the recurrence relation in Section 3, it seems difficult to compute entry $M_{i,j}$ without knowing p_i and a_i. We overcome the challenge by designing a protocol that hides the conditional testing from the client. The basic protocol is efficient and is secure in the semi-honest adversarial model.

4.1 Building Blocks

In our protocol we store values in a modularly additively split manner with a large base called L. The additively split manner means that the server and the client each has a share of a value, and the value equals to the sum of their shares modular L. If x^S and x^C represent the share of the server and the client, respectively, then the value equals to $x^S + x^C \bmod L$. We use $L - i$ to represent $-i$ (and use i to represent i). This implies that the range of the values is between $\frac{-L}{2}$ and $\frac{L}{2}$, and L must be chosen so that it is larger enough to prevent accidental wrap-around. Secure two-party private protocols were given in [28] that allow comparison of above described values. The comparison result can be additively split between the two participants, namely the server and the client. It is easy to modify these protocols to compute the maximum of the values in additively split format, which we refer to as the *private two-party maximum protocol*. We use the private two-party comparison and maximum protocols in our paper as a black box.

Our protocols use homomorphic encryption extensively. Recall that 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 [22, 42]. We utilize homomorphic encryption schemes that are semantically secure. Informally, a homomorphic scheme is *semantically secure* if the following condition holds. Given the public parameters of a homomorphic scheme E, and the encryption of one of the two messages m, m' where m is from a specific message and m' is chosen uniformly random from the message space, then $|(Pr(P(E(m))) = 1) - Pr(P(E(m')) = 1)|$ is negligible for any probabilistic polynomial time algorithm P.

4.2 Bird's Eye View of Protocol

Our basic protocol computes a $(n + 1) \times (T' + 1)$ matrix M in an additive split form. Let $M_{i,j}$ denote the value stored at the i-th row and j-th column. Let E_C be the public encryption function of the client's semantically-secure homomorphic encryption scheme. The server learns $E_C(M_{i,j})$ values for all $i \in [1, n]$ and $j \in [1, T']$. From the security of E_C, a computationally-bounded server gains no information from the $E_C(M_{i,j})$ values.

To compute the matrix we give a protocol that allows the server to compute (with the client's help) the value $E_C(M_{i,j})$ when given $E_C(M_{i',j'})$ for all values (i', j') that are dominated by (i, j), for all $i \in [1, n]$ and $j \in [1, T']$. $M_{0,j}$ and $M_{i,0}$ are zeros, for all $i \in [0, n]$ and $j \in [0, T']$.

With this protocol, the client and server can compute the matrix M. The server initializes $M_{0,j}$ to $E_C(0)$, and uses the protocol to compute the rest of the matrix together with the client. The straightforward implementation requires $O(mn)$ rounds, which can be reduced to $O(m+n)$ rounds by computing in parallel.

The private protocol should unify the operations in the two cases ($j < p_i$ and $j \geq p_i$) of the recurrence relation. Otherwise, the client can tell p_i from the computation. We solve this by designing a generic and private maximum function and by additively splitting intermediate results between the two parties.

4.3 Basic Protocol

The basic protocol is described in Figure 1. We choose to describe a protocol that is simple, rather than cluttering the exposition with minor improvements that do not improve the asymptotic performance.

When $j > T'$ (recall $T' = \sum_{i=1}^{n} p_i - T$), the server stops the protocol. The last entry $M_{n,T'}$ of the dynamic programming matrix has been computed. The client knows the marginal threshold T', as she keeps her share of the matrix. Yet, the client does not learn the individual point value p_i and access threshold T from the computation so far.

Theorem 1 *The communication complexity between the server and the client of our basic secure dynamic programming protocol is $O(nT')$, where n is the total number of credentials and T' is the marginal threshold.*

Setup: The client has published the public parameters of a semantically secure homomorphic scheme E_C. We will use the base of this scheme as the modulus for the additively split values.
Input: The server has $E_C(M_{i',j'})$ for all values (i', j') that are dominated by (i, j), where $i \in [1, n]$ and $j \in [0, T']$. The sever also has $p_1, \ldots, p_n$ and the client has $a_1, \ldots, a_n$.
Output: The server learns $E_C(M_{i,j})$.
Steps:

1. The server creates a pair of values α_0 and α_1, where $\alpha_0 = E_C(M_{i-1,j})$, and $\alpha_1 = E_C(-\infty)$ if $p_i > j$, and $\alpha_1 = E_C(M_{i-1,j-p_i})$ otherwise. Without lose of generality, we assume that a_i values defined by the client are always bounded by an integer B that is known to the server, i.e. $a_i \leq B$ for all $i \in [1, n]$. The server then uses $-B - 1$ as $-\infty$. The server also chooses two random values r_0 and r_1. It sends to the client $\alpha_0 E_C(r_0)$ and $\alpha_1 E_C(r_1)$.
2. The client decrypts the values to obtain β_0 and β_1. The server sets its shares to $-r_0$ and $-r_1$ and the client sets its shares to β_0 and $\beta_1 + a_i$. Note that the two candidate values for $M_{i,j}$ are additively split between the client and the server.
3. The client and the server engage in a private maximum protocol to compute the maximum of these two values in an additively split format, denote the shares by x^S and x^C.
4. The client sends $E_C(x^C)$ to the server, and the server computes $E_C(x^C + x^S)$ and sets this as his output.

Figure 1: Basic protocol.

Proof: n corresponds to the row of the dynamic programming table, and T' corresponds to the column of the table. Each invocation of Figure 1 fills up one cell in the table. Therefore, to fill up the entire table takes nT' invocations of Figure 1. In the basic protocol, the size of the ciphertext transmitted at each round is constant. Therefore, the theorem holds. □

The protocol presented here is secure in the semi-honest adversary model, which is improved later by our indexing expansion method in Section 6. The detailed security analysis is given in Section 7. Once the dynamic programming table has been filled, the user needs to discover the set of credentials that have been selected to disclose. This is done by traceback algorithms described next.

5 Extended Protocol I

In this section, we describe how to modify the secure protocol in Section 4 to support the back-tracking of the optimal solution (i.e., the optimal credential set to be disclosed). Therefore, the step 3 of our basic protocol is modified so that not only the maximum but also the comparison result of the two candidate values for $M_{i,j}$ are computed for all $i \in [1, n]$ and $j \in [1, T']$. During the computation, neither the server nor the user knows the result of the comparison tests, as the result is split between them.

From the recurrence relation in Section 3, it is easy to see that the comparison result directly indicates whether a_i is contained in $M_{i,j}$ and thus whether credential C_i is selected. Denote F as a matrix that contains the result of the comparisons, we modify the previous protocol so that the server learns $E_C(F_{i,j})$ for the entire matrix. The main idea of this traceback solution is that the server and the user work together to retrieve the plaintext comparison results starting from the last entry of the table, following the path of the optimal solution. We call the traceback method *traceback protocol I*. Our basic protocol for secure dynamic programming together with the traceback protocol I is referred as *extended protocol I*.

5.1 Traceback protocol I description

Figure 2 describes our traceback protocol I. To integrate traceback protocol I with the basic protocol, in the step 3 of the basic protocol, the client and the server store the result of the *comparison* (used by the maximum protocol). Then the client sends the server his share of the value encrypted with E_C. At the end of the protocol the server has $E_C(M_{i,j})$ and $E_C(F_{i,j})$ for the entire matrix.

1. The server sends the user $E_C(M_{n,T'})$. This is then decrypted with the client's private key to obtain the result $M_{n,T'}$. $M_{n,T'}$ represents the privacy score associated with the unselected credentials. If this value is acceptable to the user according to some predefined privacy standard set by the user, then the protocol continues. Otherwise, the protocol terminates.
2. The server reveals the entry $E_C(F_{n,T'})$ to the client.
3. The client $E_C(F_{n,T'})$ decrypts with her private key, and obtains $F_{n,T'} \in \{0,1\}$. The client sends the plaintext value $F_{n,T'}$ to the server (The server then knows whether C_n is selected.)
   If $F_{n,T'}$ is one, this means the credential C_n will not be disclosed. In addition, according to the recurrence relation, the corresponding entry $M_{n,T'}$ in table M is computed from entry $M_{n-1,T'}$. This indicates the next value for the server to reveal is $E_C(F_{n-1,T'})$. Otherwise, the next value to reveal to the user is $E_C(F_{n-1,T'-p_n})$. This is because entry $M_{n,T'}$ is computed from entry $M_{n-1,T'-p_n}$.
4. The revealed entries represent the path of how the optimal solution is computed. This process is repeated until n reaches zero.

Figure 2: Traceback Protocol I

The protocol assumes that the server does not maliciously alter the computation results. In the case of a malicious server, the server may send $E_C(0)$ instead of the real values to mislead the user to disclose all credentials. Although this type of attack might be caught by the user (as the user may find a subset of credentials that still satisfies the threshold constraint), we give a stronger traceback algorithm that proactively prevents this type of attacks in the next section.

Theorem 2 *The communication complexity of extended protocol I is $O(nT')$, with $O(1)$ homomorphic encryptions or decryptions at each round, where n is the total number of credentials and T' is the marginal threshold.*

Proof: n is the row of the dynamic programming table, and T' is the column of the table. Each invocation of the modified Figure 1 fills up one cell in the table. Therefore, to fill up the entire table takes nT' rounds. In traceback protocol I, each round of the communication between the server and the client discovers whether a credential C_i is selected. Therefore, $O(n)$ number of rounds are required for all the credentials. Hence, the protocol has the communication complexity of $O(nT')$. □

6 Extended Protocol II

In this section, we give a protocol for private dynamic programming computation and traceback that improves the extended protocol I in terms of security guarantees. This new approach is inspired by the *subset sum problem*, yet we stress that this solution does not require the user to solve the general subset sum problem. The main idea is to allow the user to efficiently identify the selected credentials from the optimal privacy score. The new protocol, which we refer to as *extended protocol II*, is an important step towards

a protocol that is secure against malicious servers, because it can be extended to prevent the server from tampering the computation during traceback.

In addition to solving our credential selection problem (and thus the knapsack problem), the extended protocol II can be generalized to solve the traceback problem in a large variety of integer linear programming problems. It can be used for one party to securely and privately trace the optimal solution from the final computed value, with very little or no participation from the other party. It guarantees the correctness of the traceback results, even though the other party cannot be trusted during traceback.

6.1 Extended protocol II description

Our goal is to convert the user's privacy scores $\{a_i\}$ into another set of scores $\{A_i\}$, such that the following two conditions hold. (1) The optimal credential selection computed with $\{A_i\}$ should be the same as the optimal credential selection computed with $\{a_i\}$. (2) The privacy score computed with $\{A_i\}$ should reveal which set of credentials are used to obtain that score. Thus, this transformation process requires the following two properties:

Property 1 Ordering consistency: *For two sets S and R in $2^{\{1,\ldots,n\}}$, if $\sum_{i\in S} A_i < \sum_{i\in R} A_i$, then $\sum_{i\in S} a_i \leq \sum_{i\in R} a_i$.*

Property 2 Uniqueness: *For any two distinct sets S and R in $2^{\{1,\ldots,n\}}$, $\sum_{i\in S} A_i \neq \sum_{i\in R} A_i$.*

The ordering consistency property ensures that the set of revealed credentials computed with the transformed scores is optimal even when the original scores are used. The uniqueness property guarantees that traceback is possible, as only one set of credentials can generate a specific score. Note that the above properties do not imply that an efficient traceback is possible, but our transformation leads to an efficient traceback method. We give an *indexing expansion* method that transforms a privacy score a_i to A_i as follows.

$$A_i = a_i * 2^n + 2^{i-1}.$$

In binary representation, the indexing expansion shifts the binary form of a_i to the left by n positions, and gives zeros to n least significant bits except the i-th least significant bit, which is given a one. For example, suppose there are four privacy scores 2, 3, 5, 8 or in binary form 010, 011, 101, 1000. Here n = 4. After the transformations, the expanded scores have the binary form 010 0001, 011 0010, 101 0100, 1000 1000 [1], respectively. Readers can verify that the example satisfy the two required properties. We now prove that the indexing expansion has the desired properties.

Lemma 1 *The indexing expansion achieves the ordering consistency property.*

Proof: See Appendix A.

Lemma 2 *The indexing expansion achieves the uniqueness property.*

Proof: See Appendix A.

Hence, the indexing expansion method allows the client to compute the credentials that are used to achieve a specific privacy score. Although the optimal value obtained from the secure dynamic programming with the A_i scores is different from the one with the original a_i scores, the set of credentials corresponding to the optimal privacy values are the same. We now describe the extended protocol II, which makes use of the indexing expansion.

The indexing expansion of privacy scores requires n additional bits for each credential, where n is the total number of credentials. In Theorem 3 below, we prove that in order to satisfy the uniqueness property, the number of bits required for the transformed privacy scores is bounded by $\Omega(n)$. Therefore, our indexing expansion method is efficient.

[1] The space between each binary number indicates that the last four digits come from the indexing expansion.

1. The client applies the indexing expansion to each of her privacy scores $\{a_i\}$ and obtains transformed scores $\{A_i\}$.
2. The server and the client carry out our basic dynamic programming protocol in Figure 1 of Section 4 with the transformed privacy scores $\{A_i\}$.
3. Recall that at the end of the protocol the server has computed $E_C(M^S_{n,T'})$ in the cell (n, T') of the dynamic programming matrix. The server sends $E_C(M^S_{n,T'})$ to the client.
4. The client decrypts $E_C(M^S_{n,T'})$ to obtain $M^S_{n,T'}$. She computes the optimal value by adding $M^S_{n,T'}$ with $M^C_{n,T'}$, which is her share of the $M_{n,T'}$.
5. From $M_{n,T'}$, the client can efficiently trace back the set of selected credentials. The crucial observation is that the i-th least significant bit is set to true if and only if credential i was used to obtain the value. The user writes the optimal value in its binary form, and identifies non-zero bits in the last n bits. The position of such a bit gives the index of a credential that was selected (In our dynamic programming model, this means that the credential will not be disclosed).

Figure 3: Extended Protocol II

Theorem 3 *For a transformation to satisfy the uniqueness property, the number of additional bits introduced for a privacy score is lower-bounded by $\Omega(n)$, where n is the number of credentials.*

Proof: See Appendix A.

Theorem 4 *Given the dynamic programming table computed by our basic secure protocol, the communication complexity for traceback in the extended protocol II is $O(n)$, where n is the total number of credentials; the computation cost for the server is constant, and is $O(n)$ for the user.*

Proof: See Appendix A.

Corollary 5 *For the extended protocol II, the communication complexity is $O(n^2T')$, where n is the total number of credentials and T' is the marginal threshold. The computation costs for both the server and the user are $O(n^2T')$.*

Proof: See Appendix A.

6.2 Detection of value substitution by the Server

In our above described method, although difficult, it is not impossible for a malicious server to forge its share of the optimal value and thus mislead a user to disclose more credentials. The probability of the server correctly guessing a credential's privacy score and its bit position in the indexing expansion may not be negligible. For example, the server may have $1/n$ probability of the correctly guessing the bit position of a credential, where n is the total number of credentials. And it may have $1/\max\{a_i\}$ probability of correctly guessing the privacy score, where $\{a_i\}$ represents the set of untransformed privacy scores. In Section 7, we describe a simple checksum technique for preventing the server from tampering with the traceback computation. This is done by appending randomized information to privacy scores.

7 Security

In this section, we define our security model for semi-honest adversaries and prove the security of the protocols. We then present a simple checksum technique that extends our protocols to an adversarial model stronger than semi-honest one, because it can prevent the server from tampering with the traceback computation. This technique has applications beyond the specific problem considered, and is a general method

for recovering an optimal solution from any value-computing dynamic programming computation, while detecting cheating by the participants.

We define our security model in the standard way (which is similar to the definitions given by Goldreich [31]). For more details see Appendix B.

7.1 Semi-Honest Model for the Basic Protocol

Before we prove the security of the basic protocol in Figure 1 of Section 4, we state a theorem about the security of the private two-party maximum protocol used in step 3 of the basic protocol.

Theorem 6 *The private two-party maximum protocol is secure in the semi-honest model.*

The above theorem states that there is a private two-party maximum protocol such that when given the client's inputs a^C and b^C, then if there exists an algorithm that simulates the client's view of the maximum protocol. We will call this algorithm $SIMMAX_C$ ($SIMMAX_S$) for the client (server). More specifically, if the client has input a^C and b^C, then $SIMMAX_C(a^C, b^C)$ is computationally indistinguishable from the client's view of interaction between the client and the server and the output of the protocol (i.e., the client's share of the maximum value). It is worth noting that the client's (server's) view of the max protocol is simulateable from that client's (server's) input alone, because the client's (server's) output from the maximum protocol is computationally indistinguishable from a random value.

Given such a private two-party maximum protocol, it is straightforward to show that the basic protocol in Section 4 is secure.

Theorem 7 *The basic protocol of the private two-party dynamic programming computation in the point-based trust management model is secure in the semi-honest adversarial model.*

Proof: See Appendix C.

The client's view consists of four things: i) $M_{i-1,j}+r_0$, ii) $M_{i-1,j-p_i}+r_1$ or $-\infty+r_1$, iii) the interaction from the secure two-party maximum protocol, and iv) x_C. Since r_0 and r_1 are uniformly chosen values from the server, parts i) and ii) are computationally indistinguishable from a random value. Thus the simulator outputs $(r_2, r_3, SIMMAX_C(r_2, r_3))$ for random value r_2 and r_3. □

We have shown that each individual round is secure in the above protocol. The composition follows directly from the composition theorem [14].

7.2 Semi-Honest Model for Traceback

In this section, we show that the traceback protocols are secure. Given Theorem 7, the extended protocol II in Section 6 is trivially secure, because once the server gives $E_C(M^S_{n,T'})$ to the user, the user carries out the traceback computation without communication from the server. What is left to show is that the traceback protocol I given in Section 5 is also secure.

This protocol computes a second matrix F that denotes which choices (on whether to reveal a credential) are made at each entry of the matrix. As it is a simple extension to the proof in the previous section we omit this from this proof. However, we show the traceback protocol I given in Figure 2 is secure.

Theorem 8 *The traceback protocol I is secure in the semi-honest adversarial model.*

Proof See Appendix C.

7.3 A Stronger Adversary

In this section, we show how to extend our protocol to prevent many attacks that a malicious server may attempt. Note that we do not extend the protocol to the full malicious model (which would require expensive Zero Knowledge Proofs [33]). However, we do raise the bar on common things that a malicious server may try in our model. When the server is not semi-honest, a significant problem with our protocols is that the server has $E_C(M_{i,j})$ for all matrix values. Thus, the server can replace any value of the matrix with another value $E_C(v)$ for any value v. In extended protocol II, the server has to guess the weights used for

each credential. The client can easily check if the proposed sum is created by a certain set of credentials. However, as described earlier, the server may have a non-negligible probability of successfully replacing these values. We now describe a technique that reduces the probability of a successful replacement by the server to a negligible value (in terms of a security parameter).

The basic idea is that the client uses the traceback II protocol, but with a slightly different transformation. The client creates a new set of value $\hat{A}_1, \ldots, \hat{A}_n$ that satisfy the traceback properties outlined in Section 6. For each value, A_i, the client chooses uniformly a ρ-bit value (where ρ is the security parameter), call it r_i, and it sets $\hat{A}_i = A_i 2^{\lg n+\rho} + r_i$ (where A_i is the already transformed value for traceback). It is straightforward to show that these values satisfy the properties outlined in Section 6. Furthermore, for the server to substitute a value, it would have to guess a ρ bit value, which it can guess successfully with only negligible probability in the security parameter ρ.

Another attack that the server can launch is that it can send any intermediate value of the matrix to the client, and claim that it is the final result. Because an intermediate value is well-formed, it cannot be detected by the above technique. However, the server does not gain from this type of attacks. If the server chooses a value from a higher row (with a smaller row index), then this attack can be achieved by setting the point values of some credentials to zero (i.e., they are useless to the client and are never used). If a different column is chosen, then this attack can be achieved by increasing the access threshold T. If the intermediate value is from a different row and a different column, then the effect of this attack can be achieved by increasing the threshold and setting the point values of some credentials to zero at the same time. The server may attempt to form linear combinations of row entries, but there is a non-negligible chance of being caught by the client because a repeated entry may be found.

8 Conclusions and Future Work

Our point-based trust management model recognizes the fact that different (groups of) users have different privacy requirements. It aims to address the privacy concerns of individual consumers in e-commerce transactions. We formalized the credential selection problem into a knapsack problem. We gave a semantic-secure private two-party computation protocol for solving the problem in an adversarial model that can handle cheating. The indexing expansion method that we described for the extended protocol II goes beyond the specific problem considered. It yields a general method for recovering an optimal solution from any value-computing dynamic programming computation, while detecting cheating by the participants.

The point-based trust management is an interesting framework that hosts much promising future research opportunities. One direction is to consider constraints that provide arbitrary expressiveness for specifying the privacy scores associated with any credential combinations. For example, when the privacy score for disclosing two sensitive credentials together is larger than the sum of the individual privacy scores. Another promising direction is to demonstrate how our subset-sum based traceback technique can be used to build private computing protocols for other applications such as private auction, private route planning, and secure outsourcing, etc. Last but not least, it would be interesting to study how a service provider determines the point associated with each credential. Existing reputation systems [8] can be applied to this topic, and trust management mechanisms such as delegation-based access control model [3, 4] and role-based cascaded delegation [48] can be utilized to extend the scope of trust establishment.

References

[1] M. Atallah and J. Li. Secure outsourcing of sequence comparisons. In *4th Workshop on Privacy Enhancing Technologies (PET)*, volume 3424 of *Lecture Notes in Computer Science*, pages 63–78, 2004.

[2] M. J. Atallah and W. Du. Secure multi-party computational geometry. In *Proceedings of the 7th International Workshop on Algorithms and Data Structures (WADS '01)*, volume 2125 of *Lecture Notes in Computer Science*, pages 165–179, 2001.

[3] T. Aura. On the structure of delegation networks. In *Proceedings of 11th IEEE Computer Security Foundations Workshop*, pages 14–26. IEEE Computer Society Press, 1998.

[4] T. Aura. Distributed access-rights management with delegation certificates. In *Secure Internet Programming – Security Issues for Distributed and Mobile Objects*, volume 1603 of *LNCS*, pages 211–235. Springer, 1999.

[5] M. Ben-Or and A. Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *The twentieth annual ACM symposium on theory of computing*, pages 1–10. ACM Press, 1988.

[6] E. Bertino, B. Carminati, and E. Ferrari. A temporal key management scheme for secure broadcasting of XML documents. In *ACM Conference on Computer and Communications Security (CCS 2002)*, pages 31–40, 2002.

[7] E. Bertino, E. Ferrari, and G. Mella. An approach to cooperative updates of XML documents in distributed systems. *Journal of Computer Security*, 13(2):191–242, 2005.

[8] T. Beth, M. Borcherding, and B. Klein. Valuation of trust in open networks. In *Proceedings of the Third European Symposium on Research in Computer Security (ESORICS '94)*, pages 3–18, November 1994.

[9] M. Blaze, J. Feigenbaum, and A. D. Keromytis. KeyNote: Trust management for public-key infrastructures. In *Proceedings of Security Protocols International Workshop*, 1998.

[10] D. Boneh and M. Franklin. Identity-Based Encryption from the Weil Pairing. In *Proceedings of Crypto 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 213–229. Springer, 2001.

[11] R. Bradshaw, J. Holt, and K. Seamons. Concealing complex policies with hidden credentials. In *Proceedings of 11th ACM Conference on Computer and Communications Security*, Oct. 2004.

[12] J. Camenisch and E. V. Herreweghen. Design and implementation of the *idemix* anonymous credential system. In *Proceedings of the 9th ACM conference on Computer and communications security*, pages 21–30, 2002.

[13] L. J. Camp and C. Wolfram. Pricing security. In *Advances in Information Security – Economics of Information Security*, volume 12, pages 17–34. Kluwer Academic Publishers, 2004.

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

[15] A. Chander, D. Dean, and J. C. Mitchell. Reconstructing trust management. *Journal of Computer Security*, 12(1):131–164, 2004.

[16] D. Chaum. Security without identification: transaction systems to make big brother obsolete. *Communications of the ACM*, 28(10):1030–1044, October 1985.

[17] D. Chaum, C. Crépeau, and I. Damgard. Multiparty unconditionally secure protocols. In *The twentieth annual ACM Symposium on Theory of Computing (STOC)*, pages 11–19. ACM Press, 1988.

[18] D. Chaum and J.-H. Evertse. A secure and privacy-protecting protocol for transmitting personal information between organizations. In *Proceedings of Advances in cryptology—CRYPTO '86*, pages 118–167, January 1987.

[19] D. Clarke, J.-E. Elien, C. Ellison, M. Fredette, A. Morcos, and R. L. Rivest. Certificate chain discovery in SPKI/SDSI. *Journal of Computer Security*, 9(4):285–322, 2001.

[20] 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, Dec. 2001.

[21] T. H. Cormen, C. E. Leiserson, R. L.Rivest, and C. Stein. *Introduction to algorithms*. MIT Press, 2001.

[22] 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.

[23] G. Danezis, S. Lewis, and R. Anderson. How much is location privacy worth? In *Fourth Workshop on the Economics of Information Security (WEIS 2005)*, 2005.

[24] W. Du. A study of several specific secure two-party computation problems, 2001. PhD thesis, Purdue University, West Lafayette, Indiana.

[25] D. Ferraiolo and R. Kuhn. Role-based access control. In *Proceedings of the 15th National Computer Security Conference*, 1992.

[26] M. Freedman, K. Nissim, and B. Pinkas. Efficient private matching and set intersection. In *Advances in Cryptology – Eurocrypt '04*, volume 3027 of *LNCS*, pages 1–19. Springer-Verlag, May 2004.

[27] K. Frikken, J. Li, and M. Atallah. Trust negotiation with hidden credentials, hidden policies, and policy cycles. In *To appear in NDSS 2006*, 2006.

[28] K. B. Frikken and M. J. Atallah. Privacy preserving route planning. In *WPES '04: Proceedings of the 2004 ACM workshop on Privacy in the electronic society*, pages 8–15, New York, NY, USA, 2004. ACM Press.

[29] K. B. Frikken, M. J. Atallah, and J. Li. Hidden access control policies with hidden credentials. In *Proceedings of the 3nd ACM Workshop on Privacy in the Electronic Society*, Oct. 2004.

[30] O. Goldreich. Secure multi-party computation, 2000. Working Draft.

[31] O. Goldreich. *The Foundations of Cryptography*, volume 2. Cambridge University Press, 2004.

[32] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game. In *The nineteenth annual ACM conference on theory of computing*, pages 218–229. ACM Press, 1987.

[33] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof-systems. In *STOC '85: Proceedings of the seventeenth annual ACM symposium on Theory of computing*, pages 291–304, 1985.

[34] J. E. Holt, R. W. Bradshaw, K. E. Seamons, and H. Orman. Hidden credentials. In *Proceedings of the 2nd ACM Workshop on Privacy in the Electronic Society*, Oct. 2003.

[35] Q. Huang, D. Jao, and H. J. Wang. Applications of secure electronic voting to automated privacy-preserving troubleshooting. In *Proceedings of the 12th ACM CCS*, November 2005.

[36] G. Jagannathan and R. N. Wright. Privacy-preserving distributed k-means clustering over arbitrarily partitioned data. In *Proceeding of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*, pages 593–599, 2005.

[37] L. Kissner and D. Song. Private and threshold set-intersection. In *Proceedings of CRYPTO '05*, August 2005.

[38] C. E. Landwehr. Improving information flow in the information security market. In *Advances in Information Security – Economics of Information Security*, volume 12, pages 155–163. Kluwer Academic Publishers, 2004.

[39] J. Li, N. Li, and W. Winsborough. Automated trust negotiation using cryptographic credentials. In *To appear in Proceedings of 12th ACM Conference on Computer and Communications Security (CCS)*, 2005.

[40] N. Li, J. C. Mitchell, and W. H. Winsborough. Design of a role-based trust-management framework. In *Proceedings of IEEE Symposium on Security and Privacy*, pages 114–130, 2002.

[41] H. Lipmaa. Verifiable homomorphic oblivious transfer and private equality test. In *Advances in Cryptology — Asiacrypt '03*, LNCS, pages 416–433, 2003.

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

[43] R. S. Sandhu. Lattice-based access control models. *IEEE Computer*, 26(11):9–19, November 1993.

[44] R. S. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman. Role-based access control models. *IEEE Computer*, 29, Number 2:38–47, 1996.

[45] O. S. Saydjari. Multilevel security: Reprise. *IEEE Security and Privacy*, 02(5):64–67, 2004.

[46] K. E. 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 (NDSS'01)*, February 2001.

[47] A. Shamir. Identity-based cryptosystems and signature schemes. In *Advances in Cryptology – Crypto'84*, volume 196 of *Lecture Notes in Computer Science*, pages 47–53. Springer, 1984.

[48] R. Tamassia, D. Yao, and W. H. Winsborough. Role-based cascaded delegation. In *Proceedings of the ACM Symposium on Access Control Models and Technologies (SACMAT '04)*, pages 146 – 155. ACM Press, June 2004.

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

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

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

[52] A. C. Yao. How to generate and exchange secrets. In *Proc. 27th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 162–167, 1986.

[53] M. Yokoo and K. Suzuki. Secure multi-agent dynamic programming based on homomorphic encryption and its application to combinatorial auction. In *First joint International Conference on Autonomous Agents and Multiagent Systems (AAMAS-2002)*, pages 112–119. ACM Press, 2002.

[54] T. Yu, X. Ma, and M. Winslett. PRUNES: An efficient and complete strategy for trust negotiation over the internet. In *Proceedings of the 7th ACM Conference on Computer and Communications Security (CCS-7)*, pages 210–219. ACM Press, Nov. 2000.

[55] 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.

[56] T. Yu, M. Winslett, and K. E. Seamons. Interoperable strategies in automated trust negotiation. In *Proceedings of the 8th ACM Conference on Computer and Communications Security (CCS '01)*, pages 146–155. ACM Press, Nov. 2001.

A Proofs for Traceback II Protocol

Proof of Lemma 1: For ease of notation, we use $A[S]$ to denote $\sum_{i \in S} A_i$, and $a[s]$ to denote $\sum_{i \in S} a_i$. Note that $A[S] = 2^{n+1}a[S] + \sum_{i \in S} 2^i$. Now suppose we have two sets S and R where $A[S] < A[R]$. Thus, $2^{n+1}a[S] + \sum_{i \in S} 2^i < 2^{n+1}a[R] + \sum_{i \in R} 2^i$. Now, it is easy to show that $\sum_{i \in S} 2^i < 2^{n+1}$ and $\sum_{i \in R} 2^i < 2^{n+1}$. Thus $a[S] \leq a[R]$. □

Proof of Lemma 2: To show that the sums are unique, suppose we are given two sets S and R, where $S \neq R$. There must be some element j that is in one set but not the other, without loss of generality suppose $j \in S$. Now the jth bit of $A[S]$ will be 1, but it will be 0 for $A[R]$, and thus these two values are distinct. □

Proof of Theorem 3: The following holds because of the uniqueness property:

$$\sum_{i=1}^{n} A_i \geq 2^n - 1$$

The reason for this is that: i) each subset of credentials S must have a unique privacy score, ii) there are 2^n subsets, and iii) all A_i values must be positive. This implies that the maximum A_i is at least $2^{n-\log n} - \frac{1}{n}$, because $n(2^{n-\log n} - \frac{1}{n}) = 2^n - 1$. Because the length of the maximum value is at least $n - \log n - 1$, there must exist one A_i whose length is $n - \log n - 1$. Therefore, the number of bits introduced by the transformation is lower bounded by $n - \log n - 1$, and thus is $\Omega(n)$. □

Proof of Theorem 4: Once the dynamic programming table is computed, the server only needs to send value $E_C(M^S_{n,T'})$ to the user. Hence, the number of communication rounds is constant. Because each privacy score a_i is expanded with n additional binary bits, the size of information transmitted is in the order of n – assuming that the privacy scores $\{a_i\}$ before the indexing expansion are bounded by a constant. Therefore, the communication cost of the algorithm is $O(n)$. It is trivial to show that the server's computation cost is constant. For the user, because she needs to identify $O(n)$ indexing bits, her computation cost is $O(n)$. □

Proof of Corollary 5: The proof for the communication complexity is similar to Theorem 2, except that for each round of communication, the size of ciphertext transmitted may be $O(n)$-bit longer due to the indexing expansion. Therefore, the communication complexity is $O(n^2T')$ in the extended protocol II.

For each round, both the server and the user perform constant number of homomorphic operations on transformed privacy scores $\{A_i\}$. Because A_i is $O(n)$ bits long – assuming that untransformed privacy scores $\{a_i\}$ are bounded by constant, the computation cost at each round is $O(n)$ for both parties. Hence, the computation complexities overall are $O(n^2T')$.

B Security Model

Our model is for the semi-honest adversarial model. At a high level 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 where the functionality was 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 defining security, we give a notation to represent each party's view of the protocol. Suppose A and B are the parties in our protocol, we use $view_A$ to represent all of the information that A sees during the protocol (note that this view assumes that A followed the protocol as we are in the semi-honest model). Similarly we define $view_B$ for B's view. A protocol is secure against a semi-honest A if and only if there exists a algorithm that can simulate $view_A$ when given A's inputs and A's output.

To be more precise, two probability ensembles $X \stackrel{\text{def}}{=} \{X_n\}_{n\in\mathcal{N}}$ and $Y \stackrel{\text{def}}{=} \{Y_n\}_{n\in\mathcal{N}}$ are computationally indistinguishable (i.e., a polynomial bounded algorithm cannot distinguish the two distributions) if for any PPT algorithm D, any positive polynomial p, and sufficiently large n it holds that: $|(Pr(D(X_n, 1^n) = 1)) - (Pr(D(Y_n, 1^n) = 1))| < \frac{1}{p(n)}$ Thus if we let A's input and output be represented by A_I and A_O respectively, then a protocol is secure in the semi-honest model, against a corrupt A, if there is an algorithm SIM_A such that $view_A$ and $SIM_A(A_I, A_O)$ are computationally indistinguishable (i.e., SIM_A simulates A's view of the protocol).

C Proofs of Security

Proof of Theorem 7: We must show that the server's and the client's view is simulateable from their input and output alone. The server's view consists of three things: i) the interaction from the secure two-party maximum protocol, ii) the value x^S (i.e., the server's output) from the secure max protocol, and iii) $E_C(x^C)$. The simulator for the server outputs $(SIMMAX_S(-r_0, -r_1), E_C(r_2))$ for randomly chosen values r_0, r_1 and r_2. This simulation is computationally indistinguishable from the real view because of Theorem 6 and by the semantic security properties of E_C.

Proof of Theorem 8: The server's output from this protocol is either a set of credentials that the client has disclosed or is an ABORT command from the client (when the privacy requirement is too large). Now, the server's view is simply the ABORT or whether each credential is revealed by the client. This is trivially simulateable by the server's output.

The client's output is the privacy requirement of gaining access and the set of credentials that are too be revealed to the server (if it does not abort). The client's view of the protocol is $E_C(M^S_{n,T'})$ and $E_C(F_{i,j})$ (for each row i). These values are just the output information encrypted with E_C, and thus are trivially simulateable. □