Point-Based Trust: Define How Much Privacy Is Worth*

Danfeng Yao[1], Keith B. Frikken[2], Mikhail J. Atallah[3], and Roberto Tamassia[1]

[1] Department of Computer Science, Brown University
Providence, RI 02912 USA
{dyao, rt}@cs.brown.edu
[2] Department of Computer Science and Systems Analysis
Miami University
Oxford, OH 45056 USA
frikkekb@muohio.edu
[3] Department of Computer Science, Purdue University
West Lafayette, IN 47907 USA
mja@cs.purdue.edu

Abstract. This paper studies the notion of point-based policies for trust management, and gives protocols for realizing 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 Alice access to a resource. In turn, Alice values each of her credentials with a privacy score that indicates her reluctance to reveal that credential. Bob's valuation of credentials and his threshold are private. Alice's privacy-valuation of her credentials is also private. Alice wants to find 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 protocols for computing such a subset of Alice's credentials without revealing any of the two parties' above-mentioned private information.

Keywords: Trust management, private multi-party computation, knapsack problem.

1 Introduction

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

* Portions of this work were supported by Grants IIS-0325345, IIS-0219560, IIS-0312357, and IIS-0242421 from the National Science Foundation, Contract N00014-02-1-0364 from the Office of Naval Research, by sponsors of the Center for Education and Research in Information Assurance and Security, and by Purdue Discovery Park's e-enterprise Center.

P. Ning, S. Qing, and N. Li (Eds.): ICICS 2006, LNCS 4307, pp. 190–209, 2006.

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 gives a trust management model that quantitatively addresses degrees of sensitivity. Moreover, the degree of sensitivity of a given credential is private to each user, and can vary from one user to another.

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 [5,37,38,44,45], 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 [38,43].

Although mitigating probing attacks, the requirements of the trust negotiation literature have some practical limitations. **(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)** Premature information leaking is difficult to prevent in existing trust negotiation protocols including the recent framework using cryptographic credentials [32]. The premature information leaking refers to the situation when a negotiation is not successful, however sensitive credentials 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 amount of information released from the client.

As will become clear soon, the approach presented in this paper mitigates the above-mentioned problems. The computation for determining whether a user satisfies a policy is privacy-preserving, where *neither* party needs to disclose sensitive information. Of the multiple ways of satisfying the policy, Alice will tend to use the one that utilizes the credentials whose privacy she values least.

1.2 Overview

Quantitatively addressing trust establishment problem has existed in several papers on trust and reputation models [4,17,42,46]. These models have applications in open systems such as mobile ad hoc networks, Peer-to-Peer networks [17], and e-trade systems.

We consider a new 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. The resource owner, Bob, defines an admissible 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 private and not revealed.

1.3 Applications

In the point-based model, credentials are mapped with point values defined by the resource owner, therefore the client's reward or service can be dynamically adjusted according to the amount of private information revealed. The 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.

Another important type of applications for point-based model is privacy-aware presence systems [27,39,42], where presence data such as the location of a user is collected through devices such as GPS on a cellphone. The management of presence data is crucial, because it concerns not only user privacy, but also safety: presence data can be used to track and profile individuals. In the meantime, there may be emergency situations or extenuating circumstances when certain parties (like emergency workers) should have access to this kind of information, and friends and relatives of a user might be allowed to query his or her location information at any time. Therefore, a desirable feature of a location query system is that it provides different levels of precision based on the requester's trustworthiness or the context of the query. This requires a flexible authorization model for accessing the private location data, which can be offered by the point-based authorization model.

1.4 Our Contributions

1. We propose a point-based trust management model and we formalize the credential selection problem of the model into a knapsack problem. Our point-based trust management model enables users to quantitatively distinguish the sensitivities of different credentials. It also allows a provider to quantitatively assign values to credentials held by clients. The point-based model has several features: **(i)** Policy specification is simple and easily allows dynamic adjustment of services provided based on released credentials; **(ii)** A user can proactively decide whether the potential privacy loss is worth the service without disclosing any sensitive information; **(iii)** To satisfy a policy, a user can select to disclose the *optimal* credential set that minimizes the privacy loss, based on his or her personal measure.
2. We 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 complexity of our basic protocol is $O(nT')$, where n is the total number of credentials and T' is the (private) *marginal threshold*, which corresponds to the sum of the points of the credentials that are *not* disclosed. The protocol uses homomorphic encryptions, and is semantically secure against semi-honest adversaries.

   Our improved protocol, the *fingerprint protocol*, is secure in an adversarial model that is stronger than a semi-honest one (a.k.a honest-but-curious). The improved protocol 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. The complexity of our fingerprint protocol is $O(n^2T')$.
3. 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. Using this method, a participant is not required to trust the other party during the back-tracing phase. This is possible because the participant is able to efficiently identify whether the other party has tampered with the computation. For 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.

Organization of the Paper. Our point-based trust management model is presented in Section 2. The basic protocol for privacy-preserving credential selection is given in Section 3. Fingerprint protocol is given in Section 4. We analyze the security in Section 5. We present an extension to the fingerprint protocol in Section 6. Related work is given in Section 7.

2 Model

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

2.1 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 is worth ten points. This means that a client 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*.

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 is met by the disclosed credentials. Different clients have different perspective on the sensitivity of their credentials, even though the credentials are of the same type. For example, a teenager may consider 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 client 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 privacy score 10 to her college ID, and 50 to her credit card. The client is granted access to a certain resource if the access threshold is met and all of the disclosed credentials are valid. Otherwise, the access is denied. From the requester's point of view, the central question is how to fulfill the access threshold while disclosing the *least* amount of sensitive information. In the next section, we define this as a credential selection problem. 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.

Where do point values come from? One approach to obtain point values is from reputation systems [4,36,46]. Essentially the point value of a credential represents the trustworthiness of the organization that issues the credential. If a resource owner thinks organization A is more reputable than organization B, the resource owner specifies a higher point value for a credential issued by A than the one issued by B. This idea has been explored in a recent paper that quantitatively studies the connections between computational trust/reputation models with point values in point-based trust management. The paper also discusses the application of such models in privacy-preserving location systems. The work in trust models and reputation systems [4,36,46] serve as a starting point for demonstrating the applicability of point-based trust management.

2.2 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 client) 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 $a_i \geq 0$ 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 a large integer. Thus, the (knapsack) algorithm avoids choosing that credential type, as the cost is high. The server defines T that is the *access threshold* of the requested resource. Integer $p_i \geq 0$ is the *point value* for releasing credential type C_i. The requester considers all of a_i values sensitive, and the server considers the access threshold T and all of 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:

$$\min \quad \sum_{i=1}^{n} x_i a_i$$

$$\text{subject to} \sum_{i=1}^{n} x_i p_i \geq T$$

The above minimization problem can be rewritten into a knapsack problem with a 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.

We define the marginal threshold T', which coarsely correlates to the sum of the points of the 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 [15]. 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. The dynamic programming solution is pseudo-polynomial: the running time is in $O(nT')$.

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 last table entry has 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 bookkeeping the information of where the value of an entry comes from.

For our credential selection problem, the above recurrence relation can be interpreted as follows. 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, the algorithm compares the score $M_{i-1,j}$ for not selecting the i-th credential with 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.

Once the client has selected the set of credentials to disclose, she reveals them to the server. The server then verifies the validity of the credentials by checking the credential issuers' signatures.

Privacy score of a credential set. In the current model, the privacy score of multiple credentials is the sum of each individual privacy score. The summation is simple to model, and represents the additive characteristic of privacy, i.e., the more personal information revealed, the more privacy lost. Another advantage of the summation of privacy scores is the efficiency; the specification of privacy scores has a size linear in the number of credentials. However, the client may want to explicitly specify an arbitrary privacy score of a certain group of sensitive credentials. The group privacy score may be higher or lower than the sum of individual privacy scores. The latter case can happen when one credential might subsume or include some information that is included in the other credential(s). However, the dynamic programming solution is not clear for the dynamic programming problem with arbitrary constraints. It remains an interesting open question how to formulate the dynamic programming to support arbitrary privacy score specifications.

3 Basic Protocol

We present the basic protocol, which is a secure two-party dynamic-programming protocol for computing the optimal solution of the credential selection problem. The basic protocol has two sub-protocols: recursion and traceback, which represent the two phases of dynamic programming. The protocol maintains the secrecy of sensitive parameters of both parties. Furthermore, neither the server nor the client 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 2, 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.

3.1 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 [20] that allow comparison of above described values, in which the comparison result is additively split between 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 [16,34]. 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.

3.2 Overview of Basic Protocol

The basic protocol consists of two sub-protocols: the basic recursion sub-protocol and the basic traceback sub-protocol.

– Basic recursion sub-protocol: the client and server compute 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. The server computes (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 $j \in [0,T']$.

– Basic traceback sub-protocol: once the dynamic programming table has been filled out, the client discovers (with the server's help) the set of credentials that have been selected to disclose. The optimal selection is revealed to both parties.

Note that the basic recursion sub-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 learn 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.

3.3 Basic Recursion Sub-protocol

The basic recursion sub-protocol is described in Figure 1.

When $j > T'$ (recall that $T' = \sum_{i=1}^{n} p_i - T$), the server terminates the protocol. The last entry $M_{n,T'}$ of the dynamic programming matrix has been computed. The

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 point values $p_1,\ldots,p_n$ and the client has privacy scores $a_1,\ldots,a_n$.
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 loss 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 random values r_0 and r_1, and 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 value as his output.

Fig. 1. Basic recursion sub-protocol

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.

Lemma 1. *The complexity of the basic recursion sub-protocol 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.*

The proof of Lemma 1 is in the full version of the paper [41].

The basic recursion sub-protocol runs in $O(nT')$, where marginal threshold T' or the number of credentials n can potentially be large. We point out that an important advantage of our protocol compared to conventional boolean-based policies lies in the privacy-preserving functionality offered. Our protocol not only computes the optimal selection of credentials, but also does it in a privacy-preserving fashion for both the server and client. For conventional policies, the latter aspect cannot be easily achieved without having the server to publish or disclose unfairly its policies.

The protocol presented here is secure in the semi-honest adversary model, which is improved later by our indexing expansion method in Section 4. The detailed security analysis is given in Section 5.

3.4 Basic Traceback Sub-protocol

To support the back-tracking of the optimal solution (i.e., the optimal credential set to be disclosed), the basic recursion sub-protocol needs to be modified accordingly. At step 3 in the basic recursion sub-protocol, 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 client knows the result of the comparison tests, as the result is split between them. From the recurrence relation in Section 2, 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 basic recursion sub-protocol so that the server learns $E_C(F_{i,j})$ for the entire matrix. In the basic traceback sub-protocol, the server and the client work together to retrieve the plaintext comparison results starting from the last entry of the table, following the computation path of the optimal dynamic programming solution.

Figure 2 describes the basic traceback sub-protocol.

Lemma 2. *The complexity of the basic traceback sub-protocol is $O(n)$, with $O(1)$ homomorphic decryptions at each round, where n is the total number of credentials.*

The following theorem states the overall complexity of the basic protocol.

Theorem 1. *The complexity of the basic protocol is $O(nT')$, where n is the total number of credentials and T' is the marginal threshold.*

The proof of Theorem 1 is in the full version of this paper [41].

The basic traceback sub-protocol assumes that the server does not maliciously alter the computation results. In the case of a malicious server, the server may send $E_C(0)$

Input: The server has matrix entries $\{E_C(M_{i,j})\}$ and $\{E_C(F_{i,j})\}$ encrypted with the client's public key, for all $i \in [1, n]$ and $j \in [1, T']$. The client has her private key.
Output: The client learns the optimal value of the dynamic programming computation of knapsack. The server and the client learn the optimal selection of credentials, or nothing.
Steps:

1. The server sends the client $E_C(M_{n,T'})$. The client decrypts the ciphertext 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 client according to some pre-defined privacy standard set by the client, then this sub-protocol continues. Otherwise, this sub-protocol terminates.
2. The server reveals the entry $E_C(F_{n,T'})$ to the client.
3. The client decrypts $E_C(F_{n,T'})$ to obtain $F_{n,T'} \in \{0, 1\}$. The client sends the plaintext value $F_{n,T'}$ to the server (The server then knows whether C_n is selected or not.) If $F_{n,T'} = 1$, then credential C_n will not be disclosed. $F_{n,T'} = 1$ also means that entry $M_{n,T'}$ is computed from entry $M_{n-1,T'}$. Therefore, the server next reveals $E_C(F_{n-1,T'})$ to the client. If $F_{n,T'} = 0$, then the server next reveals $E_C(F_{n-1,T'-p_n})$, as the entry $M_{n,T'}$ is computed from entry $M_{n-1,T'-p_n}$.
4. The revealed entries represent the computation path of the optimal knapsack dynamic programming solution. The above process is repeated until n reaches zero.

Fig. 2. Basic traceback sub-protocol

instead of the real values to mislead the client to disclose all credentials. Although the attack might be caught by the client (as the client 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.

4 Fingerprint Protocol

In this section, we give an alternative protocol for privacy-preserving knapsack computation. The new approach is inspired by the *subset sum problem*, yet we stress that this solution does not require the client to solve the general subset sum problem. The main idea is to allow the client (*not the server*) to efficiently identify the selected credentials from the optimal privacy score. The new protocol, which we refer to as the *fingerprint protocol*,[1] 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 fingerprint protocol 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

[1] The name is because of the similarities between fingerprinting in forensics and the indexing technique that we use to uniquely identify a subset.

little or no participation from the other party. The technique guarantees the correctness of the traceback results, even though the other party cannot be trusted during traceback.

4.1 Fingerprint Protocol Description

The key idea of the fingerprint protocol is to convert the client'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. Although the above properties do not imply that an efficient traceback is possible, our transformation leads to an efficient traceback method. Our *indexing expansion* method 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, respectively. Readers can verify that the example satisfy the two required properties. We now prove that the indexing expansion has the desired properties.

Lemma 3. *The indexing expansion achieves the ordering consistency property.*

Lemma 4. *The indexing expansion achieves the uniqueness property.*

Proofs of the above two lemmas are in the full version of this paper [41].

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 fingerprint protocol, 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 Lemma 5 below, we prove that in order to satisfy the uniqueness property, the number of bits required for the transformed privacy scores is bounded by $\Omega(n)$.

Input: The server has the marginal threshold T' and point values $p_1, \ldots, p_n$. The client has privacy scores $a_1, \ldots, a_n$.
Output: The client (*not the server*) learns the optimal selection of credentials.
Steps:

1. The client applies the indexing expansion to each of her privacy scores $\{a_i\}$ and obtains the transformed scores $\{A_i\}$.
2. The server and the client carry out the basic recursion sub-protocol (in Figure 1) with the transformed privacy scores $\{A_i\}$. Recall that at the end of the basic recursion sub-protocol, the server has computed $E_C(M_{n,T'})$ in entry (n, T') of the dynamic programming matrix.
3. The server sends the ciphertext $E_C(M_{n,T'})$ to the client.
4. The client decrypts $E_C(M_{n,T'})$ to obtain $M_{n,T'}$.
5. The client expresses the optimal value $M_{n,T'}$ in binary form and identifies the non-zero bits in the last n bits. The positions of such bits give the indices of credentials that give the optimal solution[2]. Note that the i-th least significant bit of $M_{n,T'}$ is true if and only if credential i was used to obtain the optimal value.

Fig. 3. Fingerprint protocol

Lemma 5. *For any transformation of index 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.*

Theorem 2. *The complexity of the fingerprint protocol is $O(n^2T')$, where n is the total number of credentials and T' is the marginal threshold.*

The proofs of Lemma 5 and Theorem 2 are in the full version of this paper [41].

4.2 Detection of Value Substitution by the Server

In the method described above, although difficult, it is not impossible for a malicious server to forge its share of the optimal value and thus mislead a client 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 correctly guessing the bit position of a credential, where n is the total number of credentials. Also, 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 6, 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.

5 Security

We define our security model as a semi-honest (a.k.a. honest-but-curious) model. Intuitively, this means that adversaries follow the protocol but try to compute additional

information other than what can be deduced from their input and output alone. A protocol is defined as secure if it implements a function f, such that the information learned by engaging in the protocol can be learned in an ideal implementation where the functionality is provided by a trusted oracle. This definition follows the standard definitions given by Goldreich [24] for private multi-party computation.

Let A be any one of the two parties in our protocol, we use $view_A$ to represent all of the information that A sees during the protocol. A protocol is secure against a semi-honest A, if and only if there exists an 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: $|(Pr(D(X_n, 1^n) = 1)) - (Pr(D(Y_n, 1^n) = 1))| < \frac{1}{p(n)}$. Let A's input and output be represented by A_I and A_O respectively. A protocol is secure in the semi-honest model against adversary 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).

To prove the security of the basic protocol (in Figure 1), we state a lemma about the security of the private two-party maximum protocol used in step 3 of the basic protocol.

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

The above lemma states that there exists a private two-party maximum protocol such that when given the client's inputs a^C and b^C, there is an algorithm that simulates the client's view of the maximum protocol.

Given such a private two-party maximum protocol, we show that the basic recursion sub-protocol in Section 3 is secure.

Theorem 3. *The basic recursion sub-protocol is secure in the semi-honest adversarial model.*

We have shown that each individual round is secure in the above protocol. The composition follows from the composition theorem [9].

We show the basic traceback sub-protocol (in Figure 2) is secure. Note that the basic traceback sub-protocol makes uses of a matrix F that is computed in the recurrence phase. Each entry of matrix F contains the selection decision of a credential. The computation of F is secure, which can be deduced from Theorem 3.

Theorem 4. *The basic traceback sub-protocol is secure in the semi-honest adversarial model.*

Proofs of Theorem 3 and 4 are in the full version of this paper [41].

Given Theorem 3, the fingerprint protocol (in Figure 3) is secure, because once the server gives $E_C(M_{n,T'})$ to the client, the client carries out the traceback computation without any communication from the server.

Theorem 5. *The fingerprint protocol is secure in the semi-honest adversarial model.*

6 Extension

The checksum 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 discuss an extension to fingerprint protocol that is secure against an adversary who is stronger than a semi-honest one. We consider an adversarial model as described follows. An adversary may tamper with private computation by modifying intermediate results during the protocol, which is not allowed in a semi-honest model. An adversary is curious as in a semi-honest model, in that she may store all exchanged data and try to deduce information from it. An adversary is assumed to participate and follow the protocol, which is a weaker assumption than a malicious model.

It is important to define the above adversarial model. 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*).

Note that our extension does not support a full malicious model, which would require expensive Zero Knowledge Proofs [26]. 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 the fingerprint protocol, 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 idea is that the client performs transformations on his or her privacy scores. The client creates a new set of value $\hat{A}_1, \ldots, \hat{A}_n$ that satisfy the traceback properties outlined in Section 4. For each value, A_i, the client chooses uniformly a ρ-bit value (where ρ is the security parameter), which we call r_i. The client 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 4. 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.

7 Related Work

In the access control area, the closest work to ours is the framework for regulating service access and release of private information in web-services by Bonatti and Samarati [5]. They study the information disclosure in open systems such as Internet using a language and policy approach. In comparison, we design cryptographic solutions to control and manage information exchange. In addition, we focus on solving the optimality in selecting the set of credentials to disclose. Bonatti and Samarati considered two data types in the portfolio of a user: data declaration (e.g., identity, address, credit card number) and credential. Although we only consider credentials in the description of our model, the protocols can be generalized to include data declarations as long as the server and the client agree on their specifications. In general, credentials (e.g., driver's license and credit card) contain a set of data declaration information, which is usually requested as a group. For example, the credit card number and the expiration date are usually asked for at the same time. Using credentials to represent private information may be sufficient in some cases.

Our point-based trust management model quantitatively treats memberships or credentials, which is conceptually different from most 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 from conventional access control models. The flexibility provided by the point-based model enables users to proactively protect their private information. Furthermore, thresholds specified by resource owners prevent unqualified users from accessing the resource.

Anonymous credential and idemix systems have been developed [8,10,12] to allow anonymous yet authenticated and accountable transactions between users and service providers. Together with zero-knowledge proof protocols, they can be used to prove that an attribute satisfies a policy without disclosing any other information about the attribute. The work in this paper focuses on finding the optimal credentials to disclose, and can be integrated with anonymous credential systems. A zero-knowledge proof protocol can be used when the necessary information to satisfy a policy is discovered. We can apply anonymous credential techniques to implement membership credentials in the point-based trust management model. These credentials are then used to prove user's memberships without revealing individual identity.

In hidden credentials system [7,28], when a signature derived from an identity based encryption scheme [6,14,35] 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. Frikken et al. [21] give a scheme that hides

both credentials and policies. Most recently, a protocol [22] was proposed that allows both the client and the server to define *private* access policies of their credentials.

The setup of hidden credential protocols does not allow the computation of the *optimal* selection of credentials. In addition, as explained in the recent work by Frikken, Li, and Atallah [22], the server learns whether the client obtained access or not in some environments even when hidden credential schemes are used. In this case, the server can make inferences about the client's sensitive credentials. For example, if the server's policy is *one must have top secret clearance and be a FBI agent*, then the server can deduce a significant amount of information about the client when the access control decision is made. Our proposed solution allows the client to estimate potential privacy loss without leaking any sensitive information.

We have compared the trust negotiation protocols [37,38,44,45] with our point-based trust management model in the introduction. 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 [32]. The paper presents a policy language that enables negotiators to specify authorization requirements. The research on trust negotiation that is closest to ours is by Chen, Clarke, Kurose, and Towsley [13]. They developed heuristics to find an approximation of the optimal strategy that minimizes the disclosure of sensitive credentials and policies [13]. Using their methods, when negotiation fails, premature information disclosure is still a problem. Our protocols prevent premature information leakage, because the computation does not disclose sensitive parameters. Because the selection computation is private, the minimization problem is simpler to define in our point-based model than in trust negotiation frameworks. In addition, the solution computed by our basic and fingerprint protocols, if exists, is the exact optimal solution, not an approximation.

Secure Multi-party Computation (SMC) was introduced in a seminal paper by Yao [40], 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 (or Bob) learns nothing other than what can be deduced from a (or b) and $f(a, b)$. Elegant general schemes are given in [3,11,23,25] 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 [1,18], privacy-preserving computational biology [2]. The private dynamic programming protocol given by Atallah and Li [2] 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 without using Zero-Knowledge Proofs [26]. Recently, there are also solutions for privacy-preserving automated trouble-shooting

[29], privacy-preserving distributed data mining [30], private set operations [19,31], and equality tests [33]. We do not enumerate other private multi-party computation work as their approaches significantly different from ours.

Acknowledgements

We would like to thank Nikos Triandopoulos, Seth Proctor, and Kimberly Perzel for helpful comments on an earlier version of the point-based trust management model, and Aris Anagnostopoulos for helpful hints.

References

1. 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.
2. M. J. 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.
3. 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 (STOC)*, pages 1–10. ACM Press, 1988.
4. 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.
5. P. A. Bonatti and P. Samarati. A uniform framework for regulating service access and information release on the web. *Journal of Computer Security*, 10(3):241–272, 2002.
6. 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.
7. R. Bradshaw, J. Holt, and K. Seamons. Concealing complex policies with hidden credentials. In *Proceedings of 11th ACM Conference on Computer and Communications Security (CCS)*, Oct. 2004.
8. J. Camenisch and E. Van Herreweghen. Design and implementation of the *idemix* anonymous credential system. In *Proceedings of the 9th ACM Conference on Computer and Communications Security (CCS)*, pages 21–30, 2002.
9. R. Canetti. Security and composition of multiparty cryptographic protocols. *Journal of Cryptology*, 13(1):143–202, 2000.
10. D. Chaum. Security without identification: transaction systems to make big brother obsolete. *Communications of the ACM*, 28(10):1030–1044, October 1985.
11. 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.
12. 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.
13. W. Chen, L. Clarke, J. Kurose, and D. Towsley. Optimizing cost-sensitive trust-negotiation protocols. In *Proceedings of the 24th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, volume 2, pages 1431–1442, 2005.

14. 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.
15. T. H. Cormen, C. E. Leiserson, R. L.Rivest, and C. Stein. *Introduction to algorithms*. MIT Press, 2001.
16. 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 '01)*, LNCS 1992, pages 119–136, 2001.
17. E. Damiani, S. De Capitani di Vimercati, S. Paraboschi, P. Samarati, and F. Violante. A reputation-based approach for choosing reliable resources in peer-to-peer networks. In *ACM Conference on Computer and Communications Security (CCS '02)*, pages 207–216, 2002.
18. W. Du. A study of several specific secure two-party computation problems, 2001. PhD thesis, Purdue University, West Lafayette, Indiana.
19. 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.
20. K. B. Frikken and M. J. Atallah. Privacy preserving route planning. In *Proceedings of the 2004 ACM workshop on Privacy in the Electronic Society (WPES)*, pages 8–15. ACM Press, 2004.
21. 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 (WPES)*, Oct. 2004.
22. K. B. Frikken, J. Li, and M. J. Atallah. Trust negotiation with hidden credentials, hidden policies, and policy cycles. In *Proceedings of the 13th Annual Network and Distributed System Security Symposium (NDSS)*, 2006.
23. O. Goldreich. Secure multi-party computation, Oct. 2002. Unpublished Manuscript.
24. O. Goldreich. *The Foundations of Cryptography*, volume 2. Cambridge University Press, 2004.
25. 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.
26. S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof-systems. In *Proceedings of the Seventeenth Annual ACM Symposium on Theory of Computing (STOC)*, pages 291–304, 1985.
27. M. Gruteser and D. Grunwald. Anonymous usage of location-based services through spatial and temporal cloaking. In *ACM/USENIX International Conference on Mobile Systems, Applications, and Services (MobiSys)*, 2003.
28. 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 (WPES)*, Oct. 2003.
29. Q. Huang, D. Jao, and H. J. Wang. Applications of secure electronic voting to automated privacy-preserving troubleshooting. In *Proceedings of the 12th ACM Conference on Computer and Communications Security (CCS)*, November 2005.
30. 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.
31. L. Kissner and D. Song. Private and threshold set-intersection. In *Advances in Cryptology – CRYPTO '05*, August 2005.
32. J. Li, N. Li, and W. H. Winsborough. Automated trust negotiation using cryptographic credentials. In *Proceedings of 12th ACM Conference on Computer and Communications Security (CCS)*, pages 46–57, 2005.
33. H. Lipmaa. Verifiable homomorphic oblivious transfer and private equality test. In *Advances in Cryptology — Asiacrypt '03*, LNCS, pages 416–433, 2003.

34. P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. *Advances in Cryptology – EUROCRYPT 1999*, LNCS 1592:223–238, 1999.
35. 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.
36. H. Tran, M. Hitchens, V. Varadharajan, and P. Watters. A trust based access control framework for P2P file-sharing systems. In *Proceedings of the Proceedings of the 38th Annual Hawaii International Conference on System Sciences (HICSS'05) - Track 9*, page 302c. IEEE Computer Society, 2005.
37. 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.
38. 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.
39. N. Yankelovich, W. Walker, P. Roberts, M. Wessler, J. Kaplan, and J. Provino. Meeting central: making distributed meetings more effective. In *Proceedings of the 2004 ACM Conference on Computer Supported Cooperative Work (CSCW '04)*, pages 419–428, New York, NY, USA, 2004. ACM Press.
40. A. C. Yao. How to generate and exchange secrets. In *Proceedings of the 27th IEEE Symposium on Foundations of Computer Science*, pages 162–167. IEEE Computer Society Press, 1986.
41. D. Yao, K. B. Frikken, M. J. Atallah, and R. Tamassia. Flexible, secure and private point-based trust management, November 2005. Technical Report. Brown University.
42. D. Yao, R. Tamassia, and S. Proctor. Privacy-preserving computation of trust with application to fuzzy location queries, March 2006. Brown University Technical Report.
43. T. Yu, X. Ma, and M. Winslett. PRUNES: An efficient and complete strategy for automated trust negotiation over the internet. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*, pages 210–219, November 2000.
44. 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.
45. 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.
46. C. Zouridaki, B. L. Mark, M. Hejmo, and R. K. Thomas. A quantitative trust establishment framework for reliable data packet delivery in MANETs. In V. Atluri, P. Ning, and W. Du, editors, *Proceedings of the Third ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN)*, pages 1–10. ACM, 2005.