Private Distributed Scalar Product Protocol With Application To Privacy-Preserving Computation of Trust*

Danfeng Yao[1], Roberto Tamassia[1], and Seth Proctor[2]

[1] Department of Computer Science, Brown University
Providence, RI 02912 USA
{dyao, rt}@cs.brown.edu
[2] Sun Microsystems Laboratories
Burlington, MA 01803
Seth.Proctor@sun.com

Summary. In this paper, we first present a private distributed scalar product protocol that can be used for obtaining trust values from private recommendations. Our protocol allows Alice to infer the trustworthiness of Bob based on what Alice's friends think about Bob and Alice's confidence in her friends. In addition, the private information of Alice and her friends are not revealed during the computation. We also propose a credential-based trust model where the trustworthiness of a user is computed based on his or her affiliations and role assignments. The trust model is simple to compute, yet it is scalable as it classifies large groups of users.

Key words: Private multi-party computation, trust management, location privacy

1 Introduction

Conventional access decisions in stand-alone systems are usually made based on the identity of the entity requesting a resource. By comparison, in open systems such as the Internet, this approach becomes less effective. The main reason is that there is no central authority that can make access decisions. Thus, the resource owner and the requester typically belong to different security domains administrated by different authorities and are unknown to each other. For example, Alice is holding a student credential from an organization A, but Bob, the resource owner, may know nothing about A in terms of its trustworthiness, etc. Therefore, there is a strong need for designing a flexible trust establishment model.

Reputation or trust models [7, 19] provide an open, flexible, and dynamic mechanism for trust establishment, where the requester does not belong to the resource owner. Trust models have applications in distributed systems such as peer-to-peer

* Work supported in part by the National Science Foundation under ITR grant IIS–0324846.

Please use the following format when citing this chapter:

Yao, D., Tamassia, R. and Proctor, S., 2007, in IFIP International Federation for Information Processing, Volume 238, Trust Management, eds. Etalle, S., Marsh, S., (Boston: Springer), pp. 1–16.

networks, e-commerce applications such as online auctions, or in resource-sharing systems such as Grid computing. Trust models are typically built on information such as recommendations and previous experiences of individuals. Various algorithms have been proposed to evaluate trust values [6, 30], in particular how transferred trust are computed.

In this paper, we attempt to address two aspects of computational trust models: **(1)** how to protect the privacy of personal opinions during computation, and **(2)** how to design a scalable computational trust model.

In computational trust models, the recommendations on the trustworthiness of users are usually assumed to be public. However, recommendations represent one's personal opinions of other entities, and are usually considered *sensitive*. For example, Bob has bad experiences doing business with Paul on an auction site, but, he does not want to publish his negative recommendation on Paul. Alice, who has not dealt with Paul previously, would like to use Bob and others' recommendations to evaluate Paul's trustworthiness. In the meantime, Alice has her own *private* evaluations on Bob and others, which give weights to individual recommendation (e.g., Alice knows and trusts Bob, so Bob's recommendation has a higher weight.) The problem is how to enable Alice to compute the weighted recommendation on Paul without disclosing everyone's sensitive parameters. We formalize this problem as a secure multi-party computation of scalar product, and present an efficient protocol for solving it.

This paper also describes an approach to improve the scalability of trust and reputation models. Ideally, a trust model should be able to accurately and efficiently classify a group of users. In trust management applications with a large number of users, such as Shibboleth [25], the trustworthiness of individual users becomes less important if the resource owner knows the home organization of the individual. For example, if the user is a professor from a reputable college, then he or she is likely to be trustworthy. We aim to improve the scalability of the typical grass-root approach of building trust. Our approach takes advantage of the pre-existing organizational infrastructure, in particular the credential-based administration model. The trustworthiness of an individual is deduced from her digital credentials and the issuers' trustworthiness.

1.1 Our Contributions

The contributions of this paper are summarized as follows.

1. We present a private multi-party computation protocol for computing weighted trust values. The problem is for A to infer the trust value of an unknown entity X based on what other entities think about X together with A's confidence in these entities. In a world where there is no privacy concern or there is a trusted third-party, the problem can be solved by computing the scalar product of two vectors – one vector representing A's confidence values for a set of entities, and the other vector representing recommendations of these entities on X. In real life, this information is usually considered sensitive, e.g., B may not want to disclose that he does not trust X at all, and A hopes to conceal the fact that her confidence

in B is low. Private two-party scalar product protocols are available [1, 13, 31]. However, they are not suitable for our problem, where one of the vectors in the computation is distributed among multiple entities. We design an efficient private multi-party computation protocol for scalar products where individual values of a vector can have different owners. The sensitive information of all parties is not revealed (except the final scalar product).

2. We propose a credential-based trust model for inferring trustworthiness in decentralized environments. Our credential-based trust model not only simplifies and scales the decision-making process, but also improves the reliability of computed trust scores by using role certificates. We describe how to compute trust values from multiple credentials, delegation credentials, and from peers' recommendations. Our model can also be used for computing point values in the existing point-based authorization model.
3. We also describe a location-query system for giving fuzzy location information based on the trustworthiness of the query issuer. This system is a practical application of the point-based authorization model, and demonstrates the ability to give flexible yet confident trust verdicts in open systems. Location-aware applications are made popular by the increasing deployment of sensor networks, RFID, and GPS-enabled cellphone networks.

1.2 Outline of the paper

A private multi-party computation protocol for distributed scalar products is presented in Section 2. This protocol supports efficient and privacy-preserving computation of trust values. Our credential-based trust model is introduced in Section 3. In Section 4, we describe how our trust model can be integrated with the existing point-based trust management model. In Section 5, we present an application of point-based trust management to the location query problem for sensor networks. Related work is described in Section 6. Finally, future work is given in Section 7.

2 Private Distributed Scalar Product Protocol

In this section, we define, construct, and analyze the private distributed scalar product protocol. The private distributed scalar product protocol has applications in privacy-preserving data mining problems. In Section 3.2, we show how it is used to privately compute trust values from peers' recommendations.

2.1 Definitions

In what follows, we define that all arithmetic is done in $\mathbb{Z}_m$ for some m. A private distributed scalar product protocol is to compute $X \cdot Y$, where $X = (x_1, x_2, \ldots, x_n) \in \mathbb{Z}_m^n$ and $Y = (y_1, y_2, \ldots, y_n) \in \mathbb{Z}_m^n$ are vectors of length n.

The protocol is run by l numbers of players where $1 \leq l \leq 2n$, and x_i and y_i are disjointly partitioned among the players. That is, each player knows one or more

of the elements in the vectors, and a vector is known by one and only one player. In a centralized case where $l = 1$, the problem is reduced to trivial scalar product computation. If $l = 2$, i.e. a two-party private computation problem, one can use existing private scalar product protocols [1, 13, 31]. If there are $2n$ players, each party knows only one element in X or Y. The goal of the protocol is for the players to jointly compute $X \cdot Y$ without disclosing each own's private information, i.e., x_i or y_i values. The security of the protocol can be intuitively thought of as players do not gain non-negligible knowledge of others' private information (besides the final scalar product). In particular, the property should hold even if players collude. The security of the protocol is further analyzed in Section 2.4.

For our trust model in Section 3, we are interested in a specific scenario with $n + 1$ players: Alice wants to compute the point value for an unknown entity E. She knows n entities $B_1, B_2, \ldots, B_n$, and Alice's point value for entity B_i is x_i. Each entity B_i knows entity E, and has assigned point y_i to E, respectively. Alice and $B_1, B_2, \ldots, B_n$ jointly compute $X \cdot Y$, which is given to Alice at the end of the protocol, but not to any of the B_is. We present our private distributed scalar product protocol for this special case. The protocol can be easily generalized to cases where l is anywhere between 3 and $2n$, where n is the length of the vector.

2.2 Building Blocks

Our private distributed scalar product protocol uses the homomorphic encryption scheme and a private multi-party summation protocol.

Homomorphic Encryption

A homomorphic encryption scheme has three functions $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$, where Gen generates a private key sk and a public key pk, Enc and Dec are encryption and decryption functions, respectively. The encryption function Enc is said to be homomorphic, if the following holds: $\mathsf{Enc}_{\mathsf{pk}}(x; r) \cdot \mathsf{Enc}_{\mathsf{pk}}(y; r') = \mathsf{Enc}_{\mathsf{pk}}(x + y; r \cdot r')$, where x and y denote plaintext messages and r and r' denote random strings. Another property of such a scheme is that $\mathsf{Enc}_{\mathsf{pk}}(x; r)^y = \mathsf{Enc}_{\mathsf{pk}}(x \cdot y; r^y)$. This means that a party can add encrypted plaintexts by doing simple computations with ciphertexts, without having the private key. 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 [9, 21]. We utilize homomorphic encryption schemes that are semantically secure. A homomorphic scheme is called *semantically secure* when a probabilistic polynomial-time adversary cannot distinguish between random encryptions of two elements chosen by herself.

Private Multi-Party Summation Protocol

Our protocol also uses an efficient private multi-party summation protocol, which was presented by Atallah *et al.* [2]. Their protocol is to make n parties, each with a

number V_i, cooperate to *simultaneously* find out $\sum_{i=1}^{n} V_i$ without revealing to each other anything other than the answer. To achieve this, each party chooses a random value, which is used to hide the input. The intermediate sum is additively split among the participants.

The summation protocol by Atallah *et al.* [2] is briefly described as follows. Every party i has a private value V_i. Party i chooses a random number R_i. Every party $2i$ gives to $2i+1$ his $V_{2i} + R_{2i}$, then every $2i+1$ gives to $2i$ his R_{2i+1}. Let us denote A_i as the sum $V_i + R_i$ for each party i. The odd (resp., even)-numbered parties together compute the sum $A+R$ (resp., R), where $A = \sum_{i=1}^{n} A_i$ and $R = \sum_{i=1}^{n} R_i$. Note that to compute the sum, the protocol should not let each party send his share in the clear to all other parties, which is obviously insecure. The protocol in [2] gives a non-trivial way to do this by requiring the participants to compute a randomized private sum. We refer readers to the literature for details of summation procedure. Finally, the odd (resp., even) simultaneously exchange their quantities to obtain A. We use their protocol as a black box, and refer readers to the literature for more details [2].

2.3 Protocol Description

Our private distributed scalar product protocol is shown in Figure 1. Alice's input of the protocol is a private vector X. Each party B_i (for $1 \leq i \leq n$) has a private value y_i. At the end of the protocol, the scalar product $X \cdot Y$ is learned by Alice or by every participant, where $Y = (y_1, \ldots, y_n)$.

Alice encrypts each element x_i of her vector X with her public key in homomorphic encryption. The ciphertext c_i is sent to B_i, respectively. Because B_i does not know Alice's private key, Alice's value is safe. Because of the properties of homomorphic encryption, entity B_i is able to compute the ciphertext corresponding to $x_i y_i$, even though he does not know x_i. The resulting ciphertext is w_i in Figure 1. To hide y_i, B_i computes the ciphertext w_i' corresponding to $x_i y_i - s_i$, where s_i is a random number. Alice receives ciphertext w_i' from each B_i, and computes the product of all w_i's, which is decrypted to $X \cdot Y - \sum_{i=1}^{n} s_i$. Next, all of B_is carry out a private multi-party summation protocol that computes $\sum_{i=1}^{n} s_i$. At the end of the summation protocol, every B_i learns the sum. Alice obtains the sum from B_is, and computes $X \cdot Y$ without learning the individual y_i values.

Our private distributed scalar product protocol is based on the private two-party scalar product protocol by Goethalsh *et al.* [13], where each party has a vector and the protocol outputs the scalar product result of the two vectors in a split form. That is, the scalar product result is split between the two parties, and equals to the sum of two shares. The concept of shared private computation can also be found in [1, 12]. A variant of our protocol allows all participating parties to learn the scalar product result $X \cdot Y$. Alice with S_A and all B_is, each with s_i, carry out a private multi-party summation protocol with their inputs. Our analysis is based on the protocol in Figure 1.

PRIVATE INPUTS: Private vector $X = (x_1, \ldots, x_n) \in \mathbb{Z}_m^n$ by Alice; private values y_1 by entity $B_1, \ldots, y_n$ by entity B_n, where $y_i \in \mathbb{Z}_m$ for all $i \in [1, n]$.
PRIVATE OUTPUTS: Alice learns $X \cdot Y \mod m$, where m is a public parameter.

1. Setup phase. Alice does: Generate a private and public key pair $(\mathsf{sk}, \mathsf{pk})$. Send pk to all B_i.
2. Alice does for $i \in \{1, \ldots, n\}$: Generate a random new string r_i. Send $c_i = \mathsf{Enc}_{\mathsf{pk}}(x_i; r_i)$ to B_i.
3. B_i does: Set $w_i = c_i^{y_i} \mod m$. Generate a random plaintext s_i and a random nonce r_i'. Send to Alice $w_i' = w_i \cdot \mathsf{Enc}_{\mathsf{pk}}(-s_i; r_i')$.
4. Alice does: Compute the product of ciphertext w_i's as $\Pi_{i=1}^n w_i' \mod m$. Use her private key sk to decrypt the product, and obtain the partial result $S_A = X \cdot Y - \sum_{i=1}^n s_i$.
5. All B_is, each with s_i, carry out a private multi-party summation protocol with their inputs (described in 2.2). At the end of that protocol, each B_i obtains $S_B = \sum_{i=1}^n s_i$.
6. Alice does: Obtain S_B from (any of the) B_is. Compute $X \cdot Y = S_A + S_B$.

Fig. 1. Private Distributed Scalar Product Protocol. m is a public parameter of the homomorphic encryption scheme.

Operation	Scalar Product Phase	Summation Phase	Total
Comp. (Alice)	$O(n)$ homomorphic op.	$O(1)$	$O(n)$ homomorphic op.
Comm. (Alice)	$O(n)$	$O(1)$	$O(n)$
Comp. (B_i)	$O(\log y_i)$ homomorphic op.	$O(1)$	$O(\log y_i)$ homomorphic op.
Comm. (B_i)	$O(1)$	$O(1)$	$O(1)$

Table 1. Computation (Comp.) and communication (comm.) complexities of the private distributed scalar product protocol. We denote by n the length of Alice's vector X. The logarithmic factor is due to using multiplications to compute exponentiation in step 3.

2.4 Analysis of the Protocol

The correctness of the protocol is obvious. Alice obtains from B_i (for all $i \in [1, n]$) an encryption of $x_i y_i - s_i$. Alice multiplies the n ciphertexts, and decrypts to obtain the sum $\sum_{i=1}^n x_i y_i - s_i$. Once Alice obtains $\sum_{i=1}^n s_i$, she computes $X \cdot Y = \sum_{i=1}^n x_i y_i$. The security and efficiency of our private multi-party protocol for distributed scalar product are analyzed.

The security of our private multi-party scalar product protocol is based on the security of the private two-party scalar product protocol [13] and the private multi-party summation protocol [2]. In general, the multi-party protocol among players is secure when the privacy and correctness are guaranteed for all players. It is said that a protocol protects privacy when the information that is leaked by the distributed computation is limited to the information that can be learned from the designated output of the computation [22]. In our problem, Alice's private vector X and each entity B_i's private value y_i are not leaked to each other, besides the scalar product. Note

that in almost all existing private scalar product solutions, one player can construct a system of linear equations based on the specification of the protocol, and solve it for the secret values.

Our security is in the semi-honest model, where it is assumed that all players follow the protocol, but they are also curious: that is, they may store all exchanged data and try to deduce information from it. One challenge in designing the multi-party scalar product protocol is to prevent collusions among players. In particular, during the step of summation, Alice may attempt to collude with a subset of players B_is to discover the private values of other players.

As in almost all private multi-party protocols, we assume that each party inputs his or her true private values. Providing skewed values during computation can result in inaccurate results, and wasting the computation power and bandwidth of all participants including the dishonest party. In addition, the effect of providing skewed intermediate value by a participant can be achieved by raising or lowering his or her own input. This issue is standard in multi-party protocols (both semi-honest and malicious models). Suppose A wants to compute the trustworthiness of C with help of $B_1, \ldots, B_n$, and suppose B_i is a friend of C, B_i may modify the output of the protocol by raising s_i in Figure 1. As a result, A gets a higher value for C. However, B_i can achieve the same effect by choosing a different input to begin with. Therefore, this type of attacks is not considered in multi-party protocols including ours. It is worth mentioning that once detected, this type of behaviors could be folded back into the reputation of participants, which can provide incentives for being honest during the computation.

Because of the intrinsic nature of the problems considered, even if the protocol is secure in the malicious model (discussed later), multi-party computation such as ours is still vulnerable to probing attacks. For example, if A wants to learn B_i's private value y_i, A can engage the protocol with input $X = (0, \ldots, 0, 1, 0, \ldots, 0)$ by setting only the i-th entry to be one. After the protocol A learns $X * Y = y_i$, which is the private value of B_i.

The security of our protocol is summarized in the following theorem.

Theorem 1. *Assume that* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *is a semantically secure homomorphic public-key cryptosystem. The private distributed scalar product protocol presented in this section is secure in the semi-honest model. Alice's privacy is guaranteed when for all* $i \in [1, n]$, *entity* B_i *is a probabilistic polynomial-time machine. Also, for all* $i \in [1, n]$, B_i*'s privacy is information-theoretical.*

Proof (sketch): Each entity B_i only sees a random ciphertext from Alice, for which B_i cannot guess the ciphertext. This is because of the semantic security of the homomorphic encryption scheme. Hence, B_i cannot guess Alice's value x_i.

During the summation protocol, each B_i only sees random values exchanged. Hence, B_i cannot guess the random secret s_j of B_j for all $j \neq i$.

On the other hand, Alice only sees (1) random value $x_i y_i - s_i$, (2) the sum of all s_i, and (3) the final computation scalar product $X \cdot Y$. She does not gain additional information about Y besides the final scalar product. In addition, the protocol prevents collusions among Alice and a subset D of B_is to discover private y_j value

of B_j for $B_j \notin D$, because the summation protocol guarantees that all B_is learn the sum simultaneously. Thus, Alice obtains no information about any B_i except the scalar product $X \cdot Y$, and each B_i obtains no information about Alice and entity B_j for all $j \neq i$. □

The private multi-party summation protocol is efficient, as it does not require any type of encryption schemes. The summation step does not introduce significant overhead. Details of complexities are summarized in Table 1.

Security in a malicious model Malicious adversaries, unlike semi-honest ones, can behave arbitrarily without following the protocol. They may refuse to participate the protocol, abort the protocol without finishing it, and tamper with intermediate values. Any protocol secure against honest-but-curious adversaries can be modified to a protocol that is secure against malicious adversaries using standard zero-knowledge proofs showing that all parties follow the protocol. At each step of the protocol, each party uses their transcripts and zero-knowledge proofs to convince the other parties that they have followed the protocol without cheating. We do not describe the details of how this transformation is done in this paper.

3 Credential-Based Trust Model

In this section, we present a simple credential-based trust model that is useful for the trust management in distributed environments. The main idea is to convert role-based credentials and related information into quantitative trustworthiness values of a requester, which is used for making authorization decisions. Quantitative authorization policies can allow fine-tuned access decisions instead of binary (allow or deny) verdicts, and provide more diversified access options for requesters. In addition, quantitative authorization enables providers to correlate the quality of service with the qualifications of requests (e.g., more rewards or higher resolution with higher trustworthiness). This approach utilizes and leverages existing credential and role-based management infrastructure for autonomous domains (e.g., [28, 36]) and improves the accuracy of trustworthiness prediction.

Our private multi-party scalar product protocol in the previous section can be used to compute trust values from recommendations in Section 3.2.

Terminology: In our model, we define the *administrator* of a role as the organization that creates and manages the role. If a role credential of an entity D is signed and issued by the administrator of the role, that role is said to be an *affiliated role* of D (this type of role is usually obtained through the affiliation with an organization, and thus the name). If a role credential of D is instead issued through delegation and signed by entities other than the administrator of the role, that role is called a *delegated role* of D. We define an *entity* to be an organization or an individual. An entity may issue credentials. Also, an entity may have one or more affiliated roles or delegated roles, which are authenticated by role credentials. An *affiliated role credential* is the credential for an affiliated role, and is signed by the administrator of the role. Similarly, a *delegated role credential* is the credential for proving a delegated role. A *privilege* can be a role assignment or an action on a resource. A role r administered

by entity A is denoted as $A.r$. A role defines a group of entities who are members of this role.

3.1 Definitions in Credential-Based Trust Model

A trust value in the credential-based trust model represents what an entity thinks about the trustworthiness of another entity or a role in another entity. More specifically, trust value $t(A, B)$ in the credential-based trust model represents what entity A thinks about the trustworthiness of entity B; trust value $t(A, B.r)$ in the credential-based trust model represents what entity A thinks about the trustworthiness of role $B.r$ administered by entity B. For example, a Grid Computing facility GCLab assigns trust values to types of users, such as role *professor* and role *student* in a university U, and role *researcher* from a research center C. When a user holding a certain role credential requests for access to the grid computing facility, his or her privileges are specified based on the trust value of the role. Note that the credential-based trust model is different from existing trust models that generate rating certificates, which are signed certificates of one's trustworthiness generated by one's peers [23].

Ideally, an entity A maintains a trust value for each role in organization B. For example, GCLab gives different trust value to role *student* and role *professor* in a university. Hence, a requester with a *professor* role credential may be granted a different level of access privileges from a requester with a *student* role credential.

Definition 1. *If an entity A gives a role $B.r$ in B a trust value $t(A, B.r)$, then any individual who has a valid role credential of role $B.r$ issued by B has the trust value $t(A, B.r)$.*

Trust values can be derived from previous interaction experiences and/or others' recommendations, and we focus on the latter. Deriving trust values from previous transactions usually depends on specific applications, and is not discussed in this paper. In what follows, we use *trust value of a credential* to mean the trust value of the credential issuer.

3.2 Derive Trust Value From Recommendations

We describe a *weighted average* method for an entity A to compute a trust value on entity B or role $B.r$. This computation is useful when A does not have any previous interaction experience with B or $B.r$, and A wants to combine others' opinions of B or $B.r$ in forming her trust value.

In the credential-based trust model, the *recommendation* by an entity E on B is the trust value $t(E, B)$ that E gives to B. A *confidence value* represents how much A trusts the judgement of a recommender, and is defined as the trust value of A on the recommender.

Above definitions mean that recommendations are weighted by A's confidence on the recommenders. Formally, we define the weighted average computation of trust value as follows. We denote n as the number of recommenders, and E_i represents the

i-th recommender. Let MAX_TRUST be the public upper bound of all trust values. Without loss of generality, we assume a trust value is non-negative. We assume that A has already obtained her trust values $t(A, E_1)$, $t(A, E_2)$, ..., $t(A, E_n)$ on the recommenders. We also assume that each of the recommenders E_i has formed her trust value $t(E_i, B)$ on the target entity B. (In case no one in the system knows about entity B, a default trust value can be assigned to B to indicate this situation.) The formula for computing $t(A, B)$ is shown as follows, where weight $w(A, E_i) = t(A, E_i)/\text{MAX\_TRUST}$.

$$t(A, B) = \frac{1}{n} \sum_{i=1}^{n} w(A, E_i) t(E_i, B) \tag{1}$$

Value $w(A, E_i)$ represents the weight of E_i's recommendation (trust value) on B for A. Variants of weighted average computation have been used in other reputation systems, such as ordered weighted average [32]. The above description also applies when the target to be evaluated is a role, for example $B.r$, instead of an entity.

Application of private distributed scalar product protocol. Equation (1) is useful for A only when all the trust values $t(E_i, B)$ are available. However, trust value $t(E_i, B)$ is private information of E_i, who has the incentive to hide it, especially when E_i thinks negatively about B. Similarly, A may consider her trust values $t(A, E_i)$ sensitive too. The problem is how to compute the weighted average in (1) without leaking the private information of each entity. Our protocol for private multi-party scalar product in Section 2 solves this problem and satisfies the privacy requirement.

Combining trust values for access. If a requester presents multiple role credentials, then the trust values of the credentials are to be combined. For example, one simple method is to sum the trust values. This means that the requester with multiple credentials of low trust values can gain the same access privileges as a requester with one credential of a high trust value. This combination method is intuitive and is used in point-based trust management model [35].

Delegation [4, 28, 36] is important for transferring trust in decentralized environments. Associating trust values with delegation credentials is different from role credentials because the values should not only depend on the initial credential issuer, but also the intermediate delegators's trustworthiness. Our trust model can be generalized to support delegation credentials. Due to space limit, we omit this description and refer readers to the full version of our paper.

4 Integration With Point-Based Trust Management

Our proposed private multi-party protocol and trust model are useful for general access control in a decentralized environment. In this paper, we describe how it can be used for deriving point values in the existing point-based trust management model [35], which was proposed for the privacy protection of sensitive information in open environments. We briefly introduce the point-based model next.

4.1 Point-Based Trust Management

In the point-based trust management model [35], the authorization policies of a resource owner define an *access threshold* for each of its resources. The threshold is the minimum number of points required for a requester to access that resource. For example, accessing a medical database might require fifty points. The resource owner also defines a *point value* for each type of credential, which denotes the number of points or credits a requester obtains if a type of credential is disclosed. For example, a valid ACM membership might have ten points. This means that a user can disclose his or her ACM membership credential in exchange for ten points. (This is called 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.)

Each user defines a *sensitivity score* for each of their credentials. The sensitivity score represents the unwillingness to disclose a credential. For example, Alice may give a sensitivity 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 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.

The credential selection problem here 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. A private two-party dynamic programming protocol has been proposed to solve the credential selection problem [35].

4.2 Derivation of Point Values

Existing work on point-based trust management [35] does not describe how point values can be obtained or how to systematically derive points corresponding to credentials. The credential-based trust model presented in Section 3 answers this question. Using the described methods, a resource owner computes the trust values of credential issuers and their roles. The resulting trust values are to be used as point values of a resource owner in point-based trust management.

For delegation credentials presented by a requester, a resource owner can use the trust model to compute the discounted trust value of the credential. The trust value can only be computed exactly when the delegation credential is revealed. However, this information is private to the requester in the credential selection computation in point-based trust management. To mitigate this problem, a resource owner can use an approximate trust value during the credential selection computation, and then make adjustments when credentials are exchanged later.

The credential-based trust model completes the description of an important aspect in point-based authorization. Next, we give a concrete application for point-based authorization in location-query systems.

5 Applications to Location Query Systems

Privacy is an important concern in systems that use presence and other real-time user data. Presence provides great utility, but also has the potential for abuse. Managing security and privacy preferences in these systems can be complex. One approach to protect the privacy is to apply distributed anonymity algorithms to sensor networks [16, 17]. Another type of solutions is to augment existing routing protocols to enhance source-location privacy in sensor and conventional networks [18, 27].

However, these existing solutions are not suitable for several types of applications. In many scenarios such as 911 or medical emergency, road-side emergency of a GPS-enabled vehicle, and police enforcement agents, the location information of a subject is critical, and should not be hidden or anonymous. Also for example, in distributed collaboration applications such as Meeting Central [33], being able to share presence information to trusted collaborators is desirable.

Generally, sharing presence information implies sharing sensitive personal data such as computer activity, physical location, IM status, phone use, and other real-time attributes associated with a given user. Managing the privacy of this data requires capturing the user's preferences and concerns, which are typically quite individualistic. Some users feel comfortable sharing any personal details, but most want at least some control over what is shared and with whom.

A presence system can provide a service that runs on behalf of each user, acting as that user's always-online proxy. Through this proxy, the user has ultimate control over all their associated data. The proxy is resolvable based on the user's identity, and can expose services that can be queried by other entities in the system. One such service provides presence querying.

Alice's proxy chooses access decisions through a set of domain-specific entities called advisors. Each advisor provides input on possible decision responses based on its domain of expertise (e.g., reputation, purpose of the query, context of the exchange, value of the requested data). These inputs are then aggregated to determine the overall advice about a possible response. The idea is to provide a flexible mechanism that more accurately represents a user's decision process. Our credential-based trust model and point-based authorization can be used to implement a flexible advisor system.

Alice's proxy contains her policies and preferences, including the trust values of credentials that may be used for authentication. Alice also defines the precision associated with certain trust values. For example, if the trust value of the query issuer is twenty, then she might release her location information exactly. If the trust value is five, then she might release a *fuzzy interpretation* of her location, for example, the building or city where she is currently. Phrased more concretely, if Alice's closest friend, Bob, queries about her location, a precise answer is returned. If a stranger queries her location, nothing about Alice should be disclosed.

The reputation advisor computes the trust value of each query issuer, based on their credential information. The trust value is then compared to Alice's policies, and the corresponding location result is returned. The advisors reside in Alice's proxy that is a tamper-resistant system in order to prevent the leaking of private trust values.

Note that this model makes it easy to use the trust value not just in deciding what to share, but in determining the system's confidence that the right decision is made. A high trust value represents high confidence and can be executed without bothering Alice. A low trust value represents low confidence in a decision, and if low enough, may warrant interrupting Alice to check that the right decision is being made for her. This confidence metric is then fed back into the system for use the next time a similar query from the same entity arrives, and used to provide an aggregate sense of past confidence.

For location-query systems, the main advantages of using point-based trust management as opposed to conventional access control mechanisms are the flexibility of making access control decisions with an arbitrary degree of precision and the ability to derive some simple notion of confidence. In order to achieve the same expressiveness, a boolean-based access control policy would be very inefficient, as one needs to enumerate all of the possible combinations of authorizations.

6 Related Work

Secure Multi-party Computation (SMC) was introduced in a seminal paper by Yao [34], 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 (resp. Bob) learns nothing other than what can be deduced from a (resp. b) and $f(a, b)$. Elegant general schemes are given in [5, 8, 14, 15] 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, 10], privacy-preserving computational biology [3], and private two-party dynamic programming for the knapsack problem [35]. Compared to existing private scalar product protocols [1, 13, 31], our protocol is designed for general privacy-preserving distributed scalar product computation, where vector values are distributed among multiple players. The protocol has promising applications in the information discovery of reputation systems. Our security is efficient, and is comparable to the private two-party scalar product of Goethalsh *et al.* [13].

There has been much work on the privacy-awareness for ubiquitous computing environments [16, 18, 20, 26]. An existing approach to protect the location-privacy in sensor networks is through distributed anonymity algorithms that are applied in a sensor network, before service providers gain access to the data [16]. Another category of solutions is to augment existing routing protocols to enhance source-location privacy in sensor and conventional networks [18, 27]. A more fine-grained approach for managing the access to location data is based on privacy-policies [20, 26], which is closer to our solution. Using point-based authorization, we are able to support more

flexible trust establishment mechanism without rigid boolean-based policy specifications.

Our trust model work is related to the existing work on recommendation or reputation systems [6, 19] in decentralized models. Trust evidences that are generated by recommendations and past experiences have been used for trust establishment in both ad-hoc and ubiquitous computing environments [11, 24, 29]. This type of trust evidence is flexible and straightforward to collect. The notion of uncheatable reputation was proposed in recent work by Carbunar and Sion [7], who developed a reputation mechanism that prevents untruthful reputation information using witnesses. In comparison, the main property of our trust model is the use of role-based organizational infrastructure to derive trust values, which aims to improve the scalability of trust computation.

7 Conclusions and Future Work

In this paper, we have developed a general protocol for privacy-preserving multi-party scalar product computation. This protocol can be used for peers to jointly compute a weighted trust score from *private* recommendations and *private* weights. We have also presented a simple credential-based trust model for evaluating trustworthiness based on role and delegation credentials, and recommendations. Finally, we have described the architecture of a location-query system for giving fuzzy location information based on the trust score of a requester.

There are several interesting areas to explore for future work. One is to evaluate other types of trust computation besides weighted average. For example, the ordered-weighted-average operator allows the user to weight the input values in relation to their relative ordering [32]. Another promising direction is to design private multi-party protocols for other desirable functionalities in a trust model. For example, an entity wants to find out who else in the system has a similar profile of trust values as his or her own — other entities who have similar likes and dislikes. The problem becomes how to privately compute the distance between two set of trust values according to certain metrics. As part of future works, we also plan to evaluate the effectiveness of credential-based trust model in answering fuzzy location queries. This experimentation involves an implementation of the point-based authorization model, the weighted scalar protocol computation, and the comparison tests with conventional trust models.

References

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

2. M. J. Atallah, H. G. Elmongui, V. Deshpande, and L. B. Schwarz. Secure supply-chain protocols. In *2003 IEEE International Conference on Electronic Commerce (CEC 2003)*, pages 293–302. IEEE Computer Society, 2003.
3. 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.
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 (STOC)*, pages 1–10. ACM Press, 1988.
6. 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.
7. B. Carbunar and R. Sion. Uncheatable reputation for distributed computation markets. In *Financial Cryptography and Data Security Conference (FC '06)*, 2006.
8. 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.
9. 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.
10. W. Du. A study of several specific secure two-party computation problems, 2001. PhD thesis, Purdue University, West Lafayette, Indiana.
11. L. Eschenauer, V. D. Gligor, and J. Baras. On trust establishment in mobile ad-hoc networks. In *Proceedings of the Security Protocols Workshop*, April 2002.
12. 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.
13. B. Goethals, S. Laur, H. Lipmaa, and T. Mielikäinen. On private scalar product computation for privacy-preserving data mining. In C. Park and S. Chee, editors, *ICISC*, volume 3506 of *Lecture Notes in Computer Science*, pages 104–120. Springer, 2004.
14. O. Goldreich. Secure multi-party computation, Oct. 2002. Unpublished Manuscript.
15. 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.
16. 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.
17. M. Gruteser, G. Schelle, A. Jain, R. Han, and D. Grunwald. Privacy-aware location sensor networks. In *9th USENIX Workshop on Hot Topics in Operating Systems (HotOS IX)*, 2003.
18. P. Kamat, Y. Zhang, W. Trappe, and C. Ozturk. Enhancing source-location privacy in sensor network routing. In *Proceedings of 25th International Conference on Distributed Computing Systems (ICDCS)*, 2005.
19. R. Kohlas and U. M. Maurer. Confidence valuation in a public-key infrastructure based on uncertain evidence. In *Proceedings of the Third International Workshop on Practice and Theory in Public Key Cryptography (PKC '00)*, volume 1751 of *Lecture Notes in Computer Science*, pages 93–112. Springer, 2000.

20. M. Langheinrich. A privacy awareness system for ubiquitous computing environments. In *In 4th International Conference on Ubiquitous Computing*, 2002.
21. P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. *Advances in Cryptology – EUROCRYPT 1999*, LNCS 1592:223–238, 1999.
22. B. Pinkas. Cryptographic techniques for privacy-preserving data mining. *KDD Explorations*, 4(2):12–19, 2002.
23. P. Ruth, D. Xu, B. K. Bhargava, and F. Regnier. E-notebook middleware for accountability and reputation based trust in distributed data sharing communities. In C. D. Jensen, S. Poslad, and T. Dimitrakos, editors, *iTrust*, volume 2995 of *Lecture Notes in Computer Science*, pages 161–175. Springer, 2004.
24. B. Shand, N. Dimmock, and J. Bacon. Trust for ubiquitous, transparent collaboration. *Wirel. Netw.*, 10(6):711–721, 2004.
25. Shibboleth. `http://middleware.internet2.edu/shibboleth/`.
26. E. Snekkenes. Concepts for personal location privacy policies. In *In Proceedings of the 3rd ACM Conference on Electronic Commerce (CEC)*, pages 48–57. ACM Press, 2001.
27. P. F. Syverson, D. M. Goldschlag, and M. G. Reed. Anonymous connections and onion routing. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 44–54, May 1997.
28. 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.
29. G. Theodorakopoulos and J. S. Baras. Trust evaluation in ad-hoc networks. In *WiSe '04: Proceedings of the 2004 ACM workshop on Wireless security*, pages 1–10. ACM Press, 2004.
30. 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.
31. J. Vaidya and C. Clifton. Privacy preserving association rule mining in vertically partitioned data. In *Proceedings of The 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 639–644. ACM Press, July 2002.
32. R. Yager. On ordered weighted averaging aggregation operators in multi-criteria decision making. *IEEE Transactions on Systems, Man and Cybernetics*, 18(1):183–190, 1988.
33. 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.
34. 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.
35. D. Yao, K. B. Frikken, M. J. Atallah, and R. Tamassia. Point-based trust: Define how much privacy is worth. In *Proceedings of the Eighth International Conference on Information and Communications Security (ICICS '06)*, December 2006.
36. D. Yao, R. Tamassia, and S. Proctor. On improving the performance of role-based cascaded delegation in ubiquitous computing. In *Proceedings of IEEE/CreateNet Conference on Security and Privacy for Emerging Areas in Communication Networks (SecureComm '05)*, pages 157–168. IEEE Press, September 2005.