*Regular Paper*

# Design, Analysis, and Evaluation of Mobile Agent based Privacy Protection Scheme for Multi-party Computation Problems

Md. Nurul Huda,† Eiji Kamioka††,††† and Shigeki Yamada†††

Existing cryptography-based algorithms for multi-party computation problems (MPCP) are either too complex to be used practically or applicable only to the specific applications for which they have been developed. In addition, traditional (non-cryptography-based) algorithms do not provide good privacy protection for MPCPs. This paper presents the security system design, intuitive analysis, and empirical evaluation of an agent server platform for protecting privacy in solving multi-party computation problems with traditional algorithms. We have devised the security policies for an agent server platform and showed how the enforcement of those policies can effectively protect privacy while solving many MPCPs. Experimental analysis shows that the presented agent platform security system can significantly enhance privacy protection while solving the problems with traditional algorithms.

## 1. Introduction

Generally speaking, a multi-party computation problem (MPCP) deals with computing the value of a joint function from a group of parties with their inputs. For example, in the vector dominance problem [18], Alice has a bidding vector $A$ and Bob has a price vector $B$. Both of them want to determine whether they can make a deal in buying/selling $n$ items by finding out if $A$ dominates $B$, i.e. if $A_i > B_i, \forall i \in \{1, 2, \ldots n\}$, where $n$ is the number of elements in each vector. There are different types of applications of MPCP in which privacy protection is a key requirement [4),11)].

In traditional algorithms, usually the inputs are required to be aggregated to one or more parties, who evaluate the function, i.e. some of the participants' inputs need to be disclosed to others to solve the problem. The participants of an MPCP are assumed to be collaborative, but none of them may want to disclose their private inputs to others, not even to the peer members. The privacy goal in an MPCP is to compute the function without disclosing any party's private inputs to the others.

A cryptography-based algorithm can solve this problem by keeping the input data in an encrypted form. However, existing cryptography-based generalized solutions are too complex to be used practically. Researchers have developed a few problem-specific cryptography-based algorithms for solving specific MPCPs [3),18),21)] based on the specific characteristics of the related specific problem. Even though the problem-specific solutions are more efficient than the generalized solutions, the applicability of each of these cryptography-based algorithms is limited to only the specific problem for which it has been developed [18].

Traditional (non-cryptographic) algorithms can be classified as centralized or distributed. A centralized algorithm is run by a single entity (agent) and results in a high privacy loss of the participants to the central agent, because they all need to give away all their private information to the central agent. A traditional distributed algorithm is run by multiple participants (agents) in a distributed architecture and some of the participants have to disclose part of their private inputs to the others [15),16)]. Therefore, a traditional distributed algorithm also results in privacy loss.

Multi-party computation problems are generally solved with software agents that represent the participating users. Thus, in this paper, the terms "participant" and "participating agent" have been used interchangeably.

We present the design, analysis, and experimental evaluation of the security system of an agent server platform for enhancing privacy protection while solving the MPCPs with traditional (non-cryptography based) algorithms. The *basic idea* of our privacy protection scheme is to execute the traditional/existing algorithms within a closed-door one-way agent platform, but to protect the shared private information from being disclosed from the agent platform to

† University of Dhaka
†† Shibaura Institute of Technology
††† National Institute of Informatics

unauthorized parties. In our preliminary works, the concept of closed-door one-way agent platform was introduced [8]. The experimental results on the protection from various malicious activities by the participants and the service protocol were presented in papers, Refs. 9) and 10), respectively. In this paper, we at first briefly describe the security system design of the agent platform. Then, we prove through intuitive analysis that the enforcement of our proposed security policies for the agent server platform can provide very good privacy protection for solving many MPCPs. We also show a method of achieving complete privacy protection. Finally, we evaluate our privacy protection scheme by comparing the privacy loss in our scheme with that in traditional schemes for the meeting scheduling problem [14),22)] and the vector dominance problem [18].

Since, an MPCP deals with a common problem among a group of participants, for many problems, the computational result can be expressed with a common value for all of the participants. In this paper, we take into consideration a type of MPCP in which the same computational result can be revealed to all of the participants without causing privacy loss. The MPCPs with the above assumption are not trivial but typical, and many multi-party applications are covered with the above assumptions.

The rest of the paper is organized as follows. Section 2 describes the system design, including the requirement analysis, devised security policies, and architecture for enforcing those policies. Section 3 carries out the security and privacy analysis of the agent server platform. Section 4 shows the effectiveness of our scheme by comparing the privacy loss in it with those in traditional algorithms. Section 5 discusses some further considerations, and finally, Section 6 concludes the paper.

## 2. System Design

Two or more parties want to conduct a computation based on their private inputs, but neither party is willing to disclose its own inputs to the others, not even to the peer parties or a third party. We need to find a mechanism for conducting such a computation so that no more information is revealed to a participant in the computation than what can be inferred from that participant's input and the output or the computational result.

### 2.1 Requirements Analysis

In traditional computational models, the participating agents reside at different hosts in a distributed architecture and each of them is owned, administered, and controlled by individuals. Thus, the private information, which is shared with others in the problem solving process, cannot be protected from disclosing to the users or unauthorized parties. In order to realize our basic idea (i.e., to execute the traditional/existing algorithms within a closed-door one-way agent platform, but to protect the shared private information from unauthorized parties), the information sharing should take place within a centralized architecture (i.e. an agent server platform), which can be administered for uniform control over the participants. The following considerations should be taken into account for defining the security policies and designing the security architecture for the agent server platform.

(1) The server security policy should allow the participants to access minimum system resources, which are required for their normal operations and for problem solving.
(2) The participants should not be allowed to access any system resources that could be used for creating open channels with the outside world.
(3) The system security architecture must have a policy enforcement mechanism to enforce the defined policies.
(4) The restrictions on the system resources protect the participants from sending the computation result to the users themselves. So, the system should provide a mechanism for sending the computational results to the users.
(5) It must be verified that the computational result does not contain hidden private data.
(6) The computational result should also be protected from unauthorized parties by being encrypted with a group key of the participants [23].

### 2.2 Security Policies

We call our agent server platform the isolated Closed-door One-way Platform (iCOP). Its policy consists of the resource access policies and the policies for the computational result. We define the mandatory system resources as the resources without which any process can not be executed e.g., memory and CPU. All other sys-

tem resource are considered the optional system resources.

- *Resource Access Policies:* The resource access policies allow the participating agents (1) access to the mandatory resources, such as the OS, memory, and CPU, at iCOP and (2) exchange local messages within the iCOP. However, they are not granted access to any other system resources, such as files and network sockets.
- *Policies for the Computational Result:* Two policies are enforced for the computational results (1) each participating agent $A_i$ must pass the computational result $R_i$ to a stationary agent, called the service agent and (2) the computational results passed by the participating agents must be identical, i.e., $R_i = R_j \; \forall i, j \in \{1..n\}$, where $n$ is the number of participants. This requires the participating agents to follow a pre-defined format (defined with alphabet case, decimal point precision, date format etc.) for creating the computational result.

### 2.3 Security Architecture

The agent platform security architecture has two main components (**Fig. 1**): (1) the privacy manager and (2) the service agent. The privacy manager enforces the resource access policies, which protect the participants from accessing the system resources and creating any communication channels (open or covert) with the outside world. The service agent enforces the policies for the computational result and sends the results to the users. The enforcement of the policies for the computational result protects the participating agents from using covert channels through the result sending process (explanation in Section 3.2).

### 2.4 Problem Solving Mechanism

Private information sharing should take place within the iCOP so that the shared information can be kept protected from disclosure to the outside world. For solving the multi-party computation problem, the participating mobile agents, along with their private input data, migrate into the iCOP (provided by a service provider) with proper authorization, share their private information within the iCOP through local messages amongst them, and carry out the computation within the platform (**Fig. 2**). Each user must be registered with the service provider and must sign a service contract.

Under the enforced security policies, the process of creating and sending the computational result ($R$) in iCOP has been illustrated in **Fig. 3**. Each of the participating agents computes the result, arranges it using a pre-defined format, encrypts it with the group key, and passes it to the service agent. The service agent checks and verifies whether the computational result meets the policies set forth for it and sends it to the users.

### 2.5 Reliability and Scalability Issue

A single server is subject to a single point of failure. In case of using a single server, if the server fails or somehow compromised, the whole system fails. In order to handle the fault tolerance, increase the reliability and availability of the server, and achieve better scalability, we need multiple iCOP servers. These servers make up the iCOP domain. After checking digital signatures of incoming agents, the authentication server dispatches all of the participating agents of the same application to one of the several computation servers (**Fig. 4**). Each of

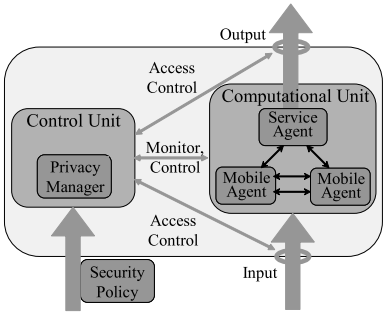



**Fig. 1** iCOP security architecture.

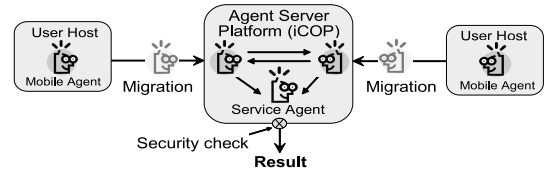



**Fig. 2** Problem solving mechanism.

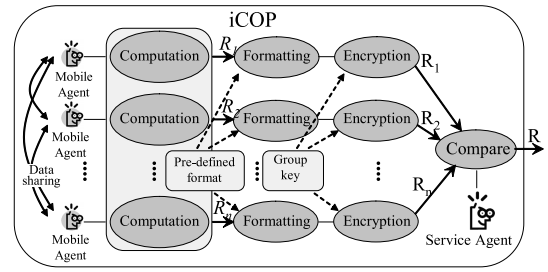



**Fig. 3** Process of creating and sending computational result in iCOP.

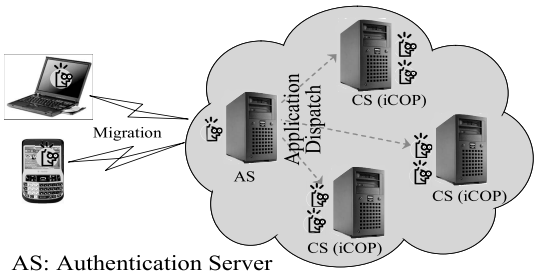



**Fig. 4** iCOP domain with multiple agent servers.

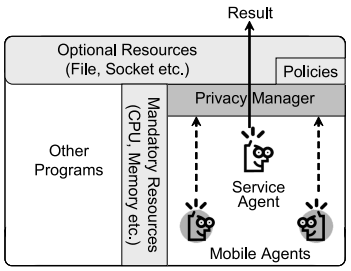



**Fig. 5** Mandatory system resources at the server are shared by all processes in the system.

the computation server of the iCOP domain must be made closed-door and one-way. By distributing different applications into different computation servers in iCOP domain, we can achieve scalability and load balancing.

If one the server systems fails, only the applications in that server will fail, the rest of the servers can continue their computation. All of the applications will not be affected by the failed system. Thus, we can achieve application-wise fault tolerance. The use of replication server or backup server also provide fault tolerance of the whole system.

### 2.6 Trust Model

The agent server must be configured according to the trusted computing specifications [12] so that the participants can trust it. A participating agent from a registered user is given authorization by checking the digital signature, which the agent carries with it. The service agent is assumed to be semi-trusted i.e., it is not malicious, would perform its functionality reliably and treats the participants equally, but private information should be kept secret from it.

## 3. Security and Privacy Analysis of Agent Server Platform

It is a common assumption in multi-party computation domains that a secure communication channel between any two hosts exists. Also, we assume that standard language level safety and operating system level safety are maintained in the server. So, because of the limited resource permissions, the participating agents cannot attack each other or the platform.

The participating agents in iCOP are not granted access to any system resources other than the mandatory resources. They cannot create any open communication channels with an outside entity by using any of the mandatory resources. However, there may be covert (hidden) channels being created that may try to leak the shared private information and all of the covert channels in a trusted system must be identified and handled [5]. Thus, we perform a covert channel analysis on iCOP.

### 3.1 Covert Channel Identification

Covert channels can be created through shared resources or objects by changing their attributes. We use the shared resource matrix method [13] to identify potential covert channels in iCOP. With this method, if an attribute of a shared resource is found that can be modified and referenced by two different processes, which are not allowed to communicate through legal channels, then potential covert channels exist through that resource.

Every process in our agent server, including those of the participating agents in iCOP, uses mandatory system resources (**Fig. 5**). A malicious participating agent in iCOP may try to modify the attributes of those resources and the malicious external processes may be able to refer to those attributes. So, potential covert channels between the participating agents and other programs in the system may exist through the mandatory system resources.

The computational result from the participating agents is sent to the outside receivers (i.e., users) by the service agent, i.e., the resulting object is shared between the participating agents and the users by transferring the object itself. Thus, potential covert channels between the participating agents and the users may exist through the computational result.

Our security policies do not allow any other objects or resources to be shared between the participants and any other entity outside the iCOP. So, there can be no other covert channels between the participating agents and the outside entity through other resources/objects. **Table 1** summarizes the data disclosure chan-

**Table 1** Summary of data disclosure channels in iCOP.

| Chan. Type | Through | Exist? |
|---|---|---|
| Open | Any resource or object | No |
| Covert | 1. Mandatory system resources | May exist |
| | 2. Computational result | May exist |
| | 3. Optional system resources | No |

nels between the participating agents in iCOP and the outside world.

### 3.2 Covert Channel Handling

Two types of covert channels may exist in iCOP: through the mandatory system resources and through the computational result (Table 1). In this section, we describe how these covert channels are handled in the iCOP.

The covert channels through the mandatory system resources are very noisy, because all the processes in the system change the attributes of the mandatory system resources. A covert channel is noisy if the corresponding shared object is available to other processes as well as to the potential data sender and receiver, and its attributes are modified by more than one process; it is noiseless if the corresponding shared object is available only to the potential data sender and the receiver and its attributes are modified by only one sender [5]. The covert channels through the mandatory system resources are common in every system, because in every system the mandatory system resources are shared among all of the processes. Thus, the underlying operating system handles these covert channels using various techniques (e.g., memory partitioning, CPU scheduling etc.) to eliminate them or to reduce their bandwidths to very low values making them ineffective [5),19]. So, we do not take any special measure for them. As an additional measures, their bandwidths can be further reduced by introducing additional delays and noise deliberately into those channels (e.g., using random allocation algorithms; introducing extraneous processes that modify covert channel variables in random patterns) [5].

The enforcement of the two specific policies (conditions) set forth for the computational results can protect the participants from using covert channels through the computational results. A covert channel through the computational result is possible only when the result is sent to the users by the service agent. Let us suppose, that an actual computational result containing no private information is $R$. Additional data can be encoded into the computational result using many different techniques [?] and the encoded data can even be kept hidden by using cryptography (**Fig. 6**). However, it must be noted that encoding any additional data into the resulting object produces a different object, regardless of the content type of the object (text or binary) and encoding method of the additional data (as shown in Fig. 6). Thus, when a participating agent $A_i$ encodes some private information with its computational result $R_i$, it will not be identical to the actual computational result $R$.

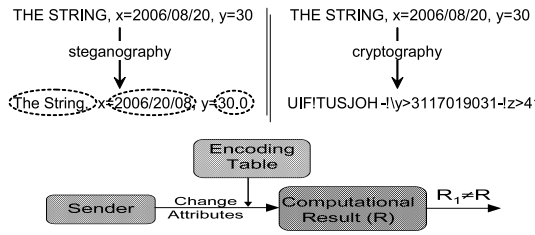



**Fig. 6** Process of encoding data into computational result by changing attributes creating different objects.

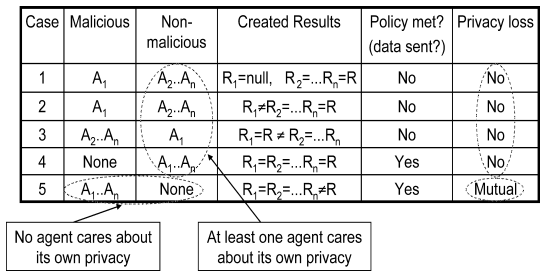



**Fig. 7** Possible cases of relations among computational results from different participants.

We categorized the possible maliciousness, associated relationships among the computational results passed by the participants, and the corresponding inferences for the enforced policies into five cases (**Fig. 7**).

In Case 1 in Fig. 7, one of the participants, say $A_1$, has not provided any computational result to the service agent. So, the policies are not met, and the result is not sent to the users, i.e. no scope for creating a covert channel. In Case 2, one of the participants, say $A_1$, is malicious, it has encoded some private information into its computational result. So, the computational results will not be identical, the policies are not met, and the result is not sent to the users, i.e. no scope for creating a covert channel. In Case 3, most of the participants are malicious and they have encoded private information with

their results, which are not identical to the non-malicious participant. Thus, the policies are not met, the computational result is not sent to the users and there is no scope for creating a covert channel. In Case 4, none of the participants are malicious, they have created identical computational results (that are equal to the actual computational result) containing no hidden information, i.e. no covert channel has been created, the policies are met, and the result is sent to the users. Finally, in Case 5, all of the participants have been considered malicious and each of them has encoded private information into their results. If they make identical results, the result is sent to the users and it causes a mutual privacy loss.

From the above analysis, we see that (mutual) privacy loss is possible when all of the participants are malicious (i.e., Case 5). If at least one of the participants cares about its own privacy and wants to protect its own private information from being leaked out, all it needs is to be not malicious and make its own result identical to the actual computational result, which eliminates the possibility of a Case 5 and there remains no scope for privacy loss. Thus, the enforcement of the policies set forth for the computational results can prevent encoding hidden data into the computational result for leakage.

### 3.3 Other Methods of Leaking Data

During our investigation, we found that it is possible to signal hidden data to the users in the result sending process without encoding the data into the computational result. We have devised one such method, which we call the result biasing method. If there are a number of possible solutions to a given problem, a protocol can be created that maps one bit stream (that is to be leaked) with each of the possible solutions numbers. For example, **Table 2** shows an example protocol in which four bit streams have been mapped with four valid solutions of the problem. The sender agent and the receiver user must use the same protocol i.e., the protocol need to be created before the agent (data sender) migrates into iCOP. To send one of the bit streams to the user, that are defined in the protocol, the potential sender agent in iCOP needs to manipulate its own inputs in the problem solving algorithm so that the computational result leads to the respective solution number in the protocol [11]. For example, with the protocol of Table 2, the sender needs to bias the result towards solution No.3 in order to send a "10" to its user and the user interprets the sent data ("10") from the sent solution number with the help of the protocol. Thus, with this method, hidden data can be sent in the result sending process without encoding it into the computational result.

**Table 2** Simple example protocol for result biasing method.

| Soln. No. | Mapped with | Soln. No. | Mapped with |
|---|---|---|---|
| 1 | 00 | 2 | 01 |
| 3 | 10 | 4 | 11 |

Leaking data through the result sending process without encoding it into the resulting object requires the existence of multiple solutions to the problem, and an agent can bias all the other agents towards a specific solution.

### 3.4 Complete Privacy

There are distributed algorithms as well as centralized algorithms for some problems. However, there are no distributed algorithms for some other types of problems (e.g., the vector dominance problem). These problems are solved with centralized algorithms. In the iCOP, even though the participants are protected from encoding the shared private information into the computational result and disclosing it to the users, they are not protected from manipulating their own inputs during execution of a distributed algorithm, and thus signaling hidden data to the users as described in Section 3.3. However, the manipulation of the inputs can be protected and complete privacy can be achieved if the inputs are exchanged first with a commitment protocol [17], and then centralized algorithms are used by each of the participants for solving the problem. A commitment protocol can detect any change in the committed values. Thus, if some of the participants manipulate the inputs (i.e., changes its inputs), the computational results created by them will not be identical to the others.

**Figure 8** shows a summary of the data disclosure channels in iCOP and their protection

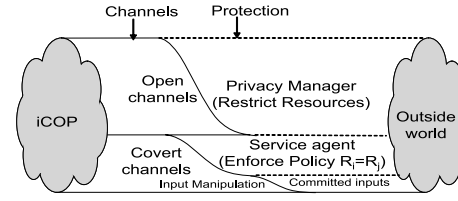



**Fig. 8** Summary of data disclosure channels in iCOP and their protection schemes.

schemes. All of the open channels are protected by the privacy manager. The possibility of creating covert channels, which requires shared resources, are protected by not sharing objects/resources, except the computational result, between the participating agents and the outside world. Enforcing the policies set forth for the computational result protects most of the scope of creating covert channels through the computational result. In addition to enforcing these policies, the participants can ensure complete privacy by committing inputs and solving the problem themselves using centralized algorithms.

## 4. Experimental Evaluation and Analysis

We evaluated our privacy protection scheme by comparing the privacy loss using the iCOP architecture with the privacy loss using the traditional architecture for the same algorithm in solving the vector dominance problem [18] and the meeting-scheduling problem [22]. For the vector dominance problem, a traditional centralized algorithm was used. For the meeting scheduling problem, we used four existing algorithms: ADOPT [16], optAPO [15], EPMS [7], and the centralized algorithm.

### 4.1 Privacy

Privacy loss was measured using the Min metric [7] and the VPS metric [14], in which information is represented in terms of states. In the Min metric, the privacy loss measure does not vary when the information is lost to more than one participant. However, in the VPS metric, the privacy loss varies based on the number of participants to whom the information is revealed.

For the vector dominance problem, the agents exchange their vectors using a commitment protocol [17] from within the iCOP, resulting in both agents getting both vectors. Then, each agent simply compares the respective components of the two vectors, creates the computational result, say a "TRUE", (in the pre-defined format) and passes it to the service agent. Even though the participants get each other's private vectors from inside the iCOP, they cannot leak anything, i.e. no privacy loss.

With the meeting scheduling problem, a number of participants schedule a meeting with their private valuations of time in each slot (i.e., preferences), whose components can take values from the set $V := \{1, \ldots, K\}$. The value of $K$ was varied from the set $\{3, 4, 5, 6, 7\}$. The number of time slots $d$ was assumed to be 10 and the number of participants $n$ was varied from 3 to 20.

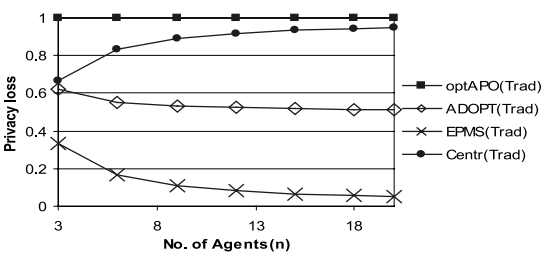
The privacy loss in optAPO occurs in the initialization phase at which the participants exchange their private information with their neighbors. The minimum privacy loss (i.e., one neighbor) in optAPO is $d/((n-1)*d) = 1/(n-1)$ in the VPS metric and $n*1/n = 1$ in the Min metric.

The privacy loss in the ADOPT algorithm varies from scenario to scenario and is calculated from the amount of information shared with the others. Thus, we measure the privacy loss in the ADOPT algorithm by taking the average privacy loss from different scenarios in the simulation.

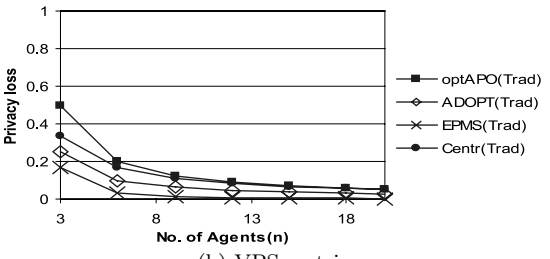
In the EPMS algorithm, the privacy loss in the VPS metric is $1/(n*(n-1))$; $n > 2$ and that in the Min metric is $d/n$ [7].

In the centralized algorithm, the privacy loss in the VPS metric is $(n-1)*(1/(n-1))/n = 1/n$ and that in the Min metric is $((n-1)*1)/n = (n-1)/n$. Note that we have taken into consideration that one of the participants solves the problem by accumulating all the inputs and using a centralized algorithm.

**Figure 9** (a) shows the privacy loss for varying numbers of participants in the four algorithms in the traditional architectures in the Min metric and Fig. 9 (b) shows the same in




(a) Min metric




(b) VPS metric

**Fig. 9** Privacy loss for four algorithms in traditional architecture.

the VPS metric. From the figure, we can see that in traditional architecture, the privacy loss in the optAPO is the highest, followed by the centralized algorithm (in which one of the participants solves the problem), the ADOPT, and the EPMS.

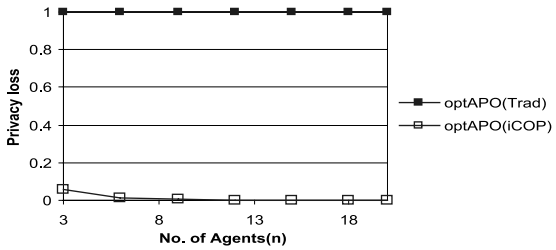
The maximum privacy loss in the iCOP depends upon the inherent privacy loss in the algorithm and the maximum amount of information that can be leaked from the iCOP. For the purpose of analysis, the maximum privacy loss in the iCOP has been shown.

**Figure 10** (a) shows the average privacy loss for optAPO in the traditional architecture and in the iCOP for the Min metric and Figure 10 (b) shows the same in the VPS metric. We see that the privacy protection by the iCOP is significant.

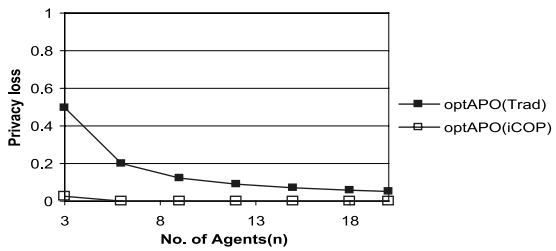
**Figures 11** and **12** show the average privacy loss in ADOPT and EPMS, respectively, and **Fig. 13** shows the average privacy loss in the traditional centralized approach and multiple centralized algorithm by each participant.

The privacy losses in the iCOP security architecture using a distributed algorithm are much lower than those the algorithm causes in the traditional distributed architecture. We took into consideration that a number of possible solutions exist and a participating agent can bias all other participating agents towards a specific solution by manipulating its own inputs. However, if there are fewer possible solutions, then the privacy loss in the iCOP architecture using a distributed algorithm will be less than that shown in Fig. 10 to Fig. 12. Finally, the privacy loss in the iCOP security architecture using a centralized algorithm executed by each participant separately (i.e., multiple centralized algorithm) with the committed inputs resulted in no privacy loss (Fig. 13).

For the distributed algorithm in the iCOP security architecture, the same information is revealed to other agents as for it in the distributed architecture, but within the iCOP and only a

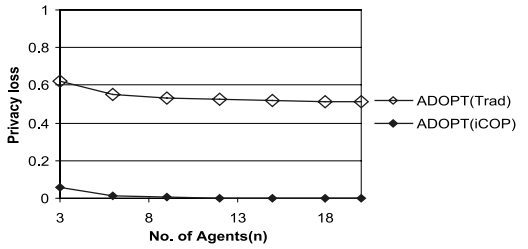


(a) Min metric

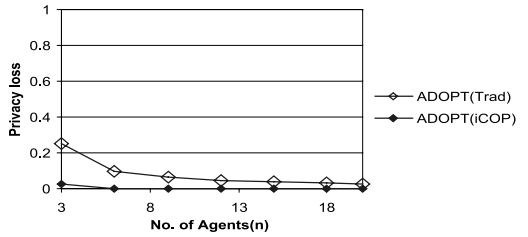


(b) VPS metric

**Fig. 10** Reducing privacy loss in optAPO algorithm using iCOP.

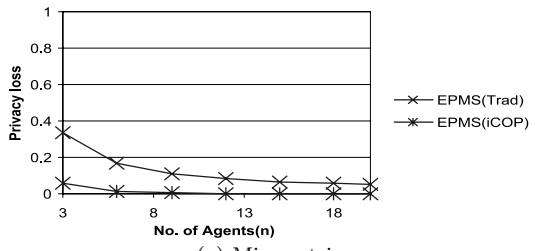


(a) Min metric

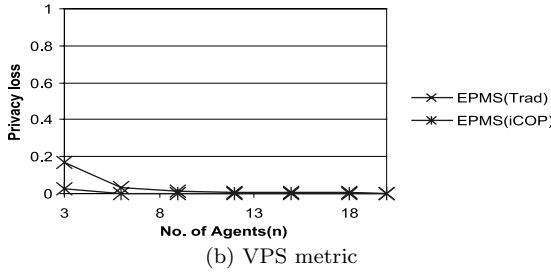


(b) VPS metric

**Fig. 11** Reducing privacy loss in ADOPT algorithm using iCOP.

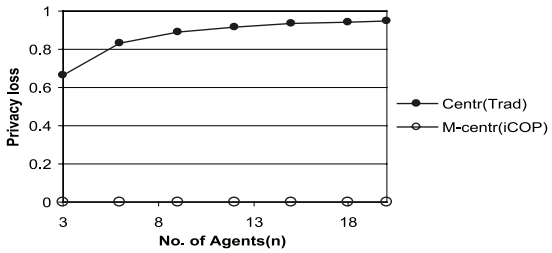


(a) Min metric

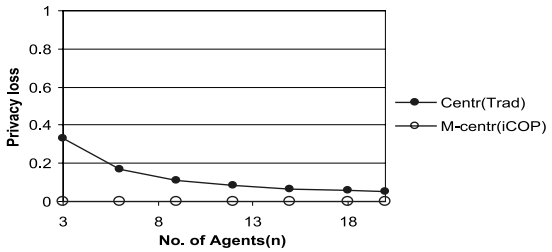


(b) VPS metric

**Fig. 12** Reducing privacy loss in EPMS algorithm using iCOP.




(a) Min metric




(b) VPS metric

**Fig. 13** Reducing privacy loss by using multiple centralized algorithms in iCOP.

very small fraction of the revealed information can be disclosed outside the iCOP, resulting in very low privacy loss. On the other hand, with each of the participants using centralized algorithms, no information can be disclosed outside the iCOP, resulting in no privacy loss.

### 4.2 Computational Time

Executing a distributed algorithm in a server may require higher computational time than executing it in a distributed architecture. Also, the execution of a centralized algorithm by multiple agents in a server will require a much higher computational time than executing it once (as in a traditional centralized scheme). For the purpose of analysis, we consider the worst case, i.e., only one agent server exists in the iCOP domain. Then, we compare the computational time in the iCOP with those in the traditional architecture for the same algorithm.

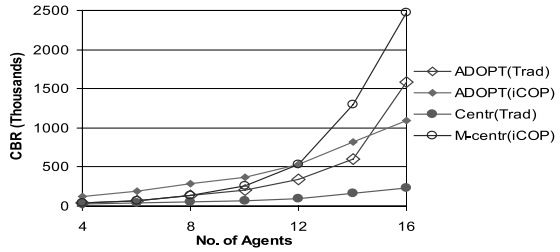
For computational time measurements, the Cycle-based Runtime (CBR) metric [2] was used, which takes into consideration the concurrency in distributed algorithms and the latency of the underlying communication environment. To measure inter-agent remote messaging delays (in traditional schemes) and interplatform agent migration delays (required in our scheme), we used two hosts (Host A and Host B) with the specifications shown in **Table 3** and the Aglet environment [24].

The computational time was measured for the ADOPT algorithm in the distributed architecture, the ADOPT algorithm in the iCOP architecture, the centralized algorithm executed by a single agent, and the centralized algorithm executed by each of the participating agents in the iCOP architecture for varying numbers of agents/participants and the averages of 15 runs have been shown in **Fig. 14** (a). For a small number of participants, the traditional centralized scheme requires the least computational time followed by the distributed algorithm in the distributed architecture, the centralized algorithm executed by each of the participants in iCOP, and the distributed algorithm in iCOP. However, with the increase in the number of agents, the total computational time increases. The growth is minimized in the centralized scheme, followed by the distributed algorithm in iCOP, the distributed algorithm in the distributed architecture, and the centralized algorithm executed by each participant in the iCOP.

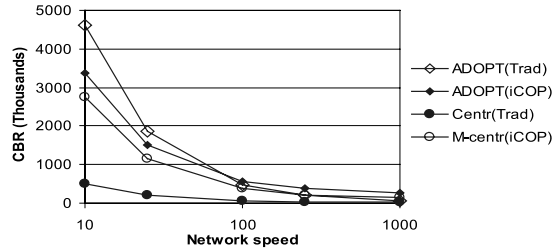
Figure 14 (b) shows the computational time (10 agents) with different relative network speeds $L$. The value of $L$ for the network used in our experiment has been labelled 100. Other

**Table 3** Hosts' specifications.

| | Host A | Host B |
|---|---|---|
| Processor | 1.4 GHz | 1.8 GHz |
| RAM | 768 MB | 256 MB |
| NIC | 100 Mbps | 100 Mbps |
| OS | Windows XP | Windows XP |




(a) For different numbers of agents.




(b) For different network speeds.

**Fig. 14** Comparison of computational times in traditional architecture and in iCOP.

values than those in $L = 100$ were estimated based upon the measured value with $L = 100$. If the communication latency is decreased i.e., if we use a faster network, say $L = 1000$, then the computational time in the distributed scheme becomes lower than that in the iCOP. However, if we use a slower network, say $L = 10$, then the computational time in the iCOP becomes lower than that in the distributed scheme. This is because a remote messaging requires quite a large computational time and a distributed algorithm in the distributed architecture requires a large number of remote messages.

## 5. Discussion

The iCOP security architecture is similar to the Java Sandbox architecture. However, the default policies of the Sandbox architecture allows it agents to connect back to the originator [6], i.e. open channels between the Sandbox and the agent originator exists. This problem cannot be solved by simply closing all the Sandbox channels, because if this is done, the computational result cannot be sent to the users. In addition, the necessary message permission for the participating agents also creates additional permissions (e.g., socket permission). Due to the additive nature of Java permissions [6], when message permission is granted to the participating agents, access to the network sockets cannot be prevented in the default Sandbox. The iCOP privacy manager explicitly checks the permission for every resource. Thus, it can precisely control (restrict) the access to every resource. In addition, the service agent and the security policies of our protection scheme help in sending only the computational result to the users, protecting the shared private information.

The traditional centralized scheme for solving the multi-party computation problem requires disclosing all the private inputs of the participants and the computational result to the central agent, who solves the problem on behalf of the participants. On the other hand, in our scheme, the participants solve the problem by themselves without disclosing their private information to a central agent and they can keep their private information from being disclosed to others. One key advantage of using the proposed protection scheme is that existing algorithms can be used for solving the problems.

Unlike cryptographic algorithms, the proposed protection scheme can be used for protecting privacy in different types of multi-party computation problems. For example, besides the problems discussed earlier in this paper, it can also be used in solving several other problems, including the permutation problem, the scalar product problem, the scientific computation problem, the geometric computation problem, and the statistical analysis problem [4].

## 6. Conclusion

We have presented the design, analysis, and experimental evaluation of a new privacy protection scheme in solving multi-party computation problems without using complex cryptographic algorithms. In the proposed scheme, the participating agents can use existing (distributed or centralized) algorithms for solving the problems, but in a confined agent platform named iCOP. In using a distributed algorithm, only a few bits may be leaked through the result sending channel depending upon the problem characteristics. The maximum privacy loss in using a distributed algorithm in our scheme is much lower than the privacy loss caused by the algorithms in a traditional distributed architecture. Also, our scheme can provide complete privacy if the participants exchange their inputs with a commitment protocol and each of them solves the problem independently using centralized algorithms with the committed inputs in the iCOP. However, this requires higher computational time than for a distributed algorithm when the number of participants is large.

## References

1) Bennett, K.: Linguistic steganography: Survey, analysis, and robustness Concerns for hiding information in text, CERIAS Technical Report 2004-313 (2004).
2) Davin, J. and Modi, P.J.: Impact of problem centralization in distributed constraint optimization algorithms, *Proc. AAMAS'05*, Netherlands, pp.1057–1063 (2005).
3) Du, W. and Atallah, M.J.: Privacy-preserving cooperative scientific computations, *Proc. 14th IEEE Workshop on Computer Security Foundations*, Canada, pp.273–282 (2001).
4) Du, W.: *A study of several specific secure two-party computation problem*, Ph.D. thesis, Purdu University (2001).
5) Gligor, V.D.: *A guide to understanding covert channel analysis of trusted systems.* http://www.fas.org/irp/nsa/rainbow/tg030.htm
6) Gong, L., Ellison, G. and Dageforde, M.: *Inside Java 2 Platform Security: Architecture,*

*API Design, and Implementation*, 2nd edition, Addison-Wesley (2003).
7) Huda, M.N., Kamioka, E. and Yamada, S.: An efficient and privacy-aware meeting scheduling scheme using common computational space, *IEICE Trans. Inf. & Syst.*, Vol.E90-D, No.3, pp.656–667 (2007).
8) Huda, M.N., Kamioka, E. and Yamada, S.: Privacy Protection in Mobile Agent based Service Domain, *Proc. ICITA'05*, Sydney, Australia, pp.482–487 (2005).
9) Huda, M.N., Kamioka, E. and Yamada, S.: Privacy Protection in Multi-Agent based Applications, *Proc.8th Intl.Conf.on Computer and Information Technology* (*ICCIT'05*), Dhaka, Bangladesh, pp.728–733 (2005).
10) Huda, M.N., Kamioka, E. and Yamada, S.: Privacy Preserving Services for Wireless Network Environments, *Proc. Intl. Conf. on Next-Generation Wireless Systems* (*ICNEWS'06*), Bangladesh, pp.166–170 (2006).
11) Huda, M.N.: *A Mobile Agent-based Privacy Protection Mechanism in Solving Multi-party Computation Problems*, Ph.D. thesis, The Graduate University for Advanced Studies (2007).
12) Kay, R.L.: How to implement trusted computing: A guide to tighter enterprise security, Technical Report, Trusted computing group.
13) Kemmerer, R.A.: A practical approach to identifying storage and timing channels: twenty years later, *Proc. ACSAC'02*, pp.109–118 (2002).
14) Maheswaran, R.T., Pearce, J.P., Bowring, E., Varakantham, P. and Tambe, M.: Privacy loss in distributed constraint reasoning: a quantitative framework for analysis and its applications, *J. of AAMAS*, Vol.13, No.1, pp.27–60, Springer (2006).
15) Mailler, R. and Lesser, V.: Solving distributed constraint optimization problems using cooperative mediation, *Proc. AAMAS'04*, New York, pp.438–445 (2004).
16) Modi, P.J., Shen, W., Tambe, M. and Yokoo, M.: ADOPT: Asynchronous distributed constraint optimization with quality guarantees, *Artificial Intelligence Journal*, Vol.161, pp.149–180 (2005).
17) Roca, J.C. and Ferrer J.D.: A non-repudiable bitstring commitment scheme based on a public-key cryptosystem, *Proc. ITCC'04*, Las Vegas (2004).
18) Sang, Y., Shen, H. and Zhang, Z.: An efficient protocol for the problem of secure two-party vector dominance, *Proc. PDCAT'05*, Dalian, China (2005).
19) Snow, B.: Four ways to improve security, *IEEE Security & Privacy Magazine*, Vol.3, No.3, pp.65–67 (2005).
20) Shooman, M.L.: *Reliability of Computer Systems and Networks — Fault Tolerance Analysis and Design*, Wiley (2002).
21) Yao, A.: Protocols for secure computations, *Proc. 23rd Annual IEEE Symposium on Foundations of Computer Science* (1982).
22) Yokoo, M., Suzuki, K. and Hirayama, K.: Secure distributed constraint satisfaction: Reaching agreement without revealing private information, *Artificial Intelligence*, Vol.161, No.1-2, pp.229–245 (2005).
23) Zhu, S., Setia, S. and Jajodia, S.: Performance optimizations for group key management schemes, *Proc. ICDCS'03*, pp.163–171 (2003).
24) http://sourceforge.net/projects/aglets/

(Received January 22, 2007)
(Accepted April 6, 2007)
(Online version of this article can be found in the IPSJ Digital Courier, Vol.3, pp.320–331.)

**Md. Nurul Huda** is an Assistant Professor at the University of Dhaka. He received his B.Sc. degree in Applied Physics and Electronics and M.Sc. degree in Computer Science from the University of Dhaka, Bangladesh in 1995 and 1997, respectively and Ph.D. degree from the Graduate University for Advanced Studies, Japan in 2007. He carried out research work at the National Institute of Informatics (NII), Tokyo from April 2004 to March 2007. His current research interests include privacy protection in multi-party computation problems and routing protocols in wireless ad-hoc networks. He is a member of the IEEE.

**Eiji Kamioka** is an Associate Professor at Shibaura Institute of Technology. He received his B.S., M.S., and Ph.D. degrees in physics from Aoyama Gakuin University in 1989, 1991, and 1997, respectively. He worked for SHARP communication laboratories from 1991 to 1993, researching and developing multimedia communication systems, and joined the Institute of Space and Astronautical Science (ISAS) in 1997 until 1998 as a Research Fellow of the Japan Society for the Promotion of Science (JSPS). Then, He proceeded to NII as an Assistant Professor in 1998. His current interests encompass ubiquitous computing networks and context-aware computing networks. He is a member of the IEICE, IEEE, IPSJ and JPS (The Physical Society of Japan).

**Shigeki Yamada** is a Professor at the National Institute of Informatics (NII). He received his B.E., M.E., and Ph.D. degrees in electronic engineering from Hokkaido University in 1972, 1974, and 1991, respectively. He worked in the NTT laboratories from 1974 to 1999, where he was involved in research and development on high-performance digital switching systems and network-wide distributed systems. He moved to NII in 1999. His current research interests include ubiquitous and context-aware computing networks, mobile networks, and privacy-enhancing technologies. From 1981 to 1982, he was a visiting scientist in the Computer Science Department, University of California, Los Angeles. He is a member of IEICE, IPSJ and a senior member of IEEE.