A Mobile Agent based Computing Model for Enhancing Privacy in Multi-party Collaborative Problem solving

Md. Nurul Huda[†], Eiji Kamioka[‡], and Shigeki Yamada[‡]

[†]The Graduate University for Advanced Studies
[‡]National Institute of Informatics,
2-1-2, Hitotsubashi, Chiyoda-ku, Tokyo 101-8430, Japan
[†]huda@grad.nii.ac.jp, [‡]{kamioka, shigeki}@nii.ac.jp

ABSTRACT

Privacy is an important issue in multi-party collaborative problems that are conventionally dealt with distributed algorithms. To get the solution, the participants need to share their private information, resulting in privacy loss. Several multi-agent algorithms try to reduce privacy loss by reducing the amount of shared private information among the agents. To address the privacy issue in multi-party collaborative computing, we propose a mobile agent based computing model and the architecture and functional components of a trusted agent, which is used in the proposed model. The participating mobile agents are trapped into the agent server, which is controlled by a service provider. The agents can interact locally from within the agent server platform and can utilize any algorithm to solve their problem, but are restricted to disclose any data to the outside world. A trusted service agent along with the participating agents carries out security analysis on the computational result to detect hidden personal data in it, if any, and sends the signed computational result to the users. Analytical result shows the effectiveness of the proposed mobile agent based computing model in terms of privacy protection and also shows the computational cost as compared with conventional distributed model.

Keywords: Mobile agent, multi-party collaborative computing, privacy, steganography, ubiquitous computing.

1 INTRODUCTION

Ubiquitous computing hides the presence of computing systems by reducing the necessity of interactiveness. Software agent technology plays a very important role in making the system autonomous. Autonomy is more important especially in multi-party collaborative problems, such as the distributed constraint satisfaction (DisCSP) problem [15][16] and distributed constraint optimization (DCOP) problem [10][11], that otherwise is a tedious and time-consuming job for human. The participants solve a common problem, which involves private data from the participants. They need to share their private valuation of certain variables to resolve the conflicts or contradictions among them in DisCSPs and also to optimize the solution in DCOPs. Existing algorithms themselves cannot offer very good privacy, which might be important in many contexts such as private event scheduling problem (e.g., meeting scheduling [7][9][16]).

Reducing privacy loss is one of the key motivating factors in many distributed algorithms. Traditionally DisCSPs and DCOPs are dealt with software agents, which act and take decisions on behalf of their users. One reason for solving a DisCSP in a distributed fashion is that the agents might not want to communicate all their private information to a central agent [15]. In the centralized approach, the privacy loss of the participants to the central agent is very high [7]. Different distributed algorithms have different level of privacy loss. Also, depending upon the metrics used for privacy loss measurement, the relative privacy loss in an algorithm may vary with respect to other algorithms [7][9].

One approach to provide privacy in DisCSPs has been to use cryptographic techniques that incurs large overhead and require to use multiple external servers which may not always be justifiable for its benefit [16]. Other cryptographic approaches have also been proposed that are very complex and limited to two-party and few specific function evaluation [4][14].

This paper proposes a new idea for enhancing privacy in multi-party collaborative computing. In the proposed computing model, the participating mobile agents take users' private data and then, along with the private data, migrate into a trusted agent server (provided by a third-party service provider) into which the mobile agents are trapped. To perform the desired computation, the agents interact with one another locally and negotiate by sharing their private data like the conventional distributed model. We describe the architecture of the agent server used in our model, its functional units, service protocol, and implementation issues. We also investigate the data disclosure channels and their protection mechanism and describe how we can achieve better privacy with the proposed model. The privacy manager of the proposed architecture restricts the participating agents from disclosing the acquired private data of other agents to the outside world. It also restricts the participating agents from leaving the platform with the shared private data. A trusted service agent sends the signed computational result to the users after the required verification that the result does not contain any hidden data in it. The verification is done by the service agent with the cooperation of the participating agents. Finally, in order to destroy the shared data, the participating agents along with their acquired data are disposed (or killed) at the agent server. We also evaluate the proposed mobile agent based model by comparing the privacy loss and computational cost for the same algorithm in

it with those in the traditional distributed model.

2 MULTI-PARTY COLLABORATIVE COMPUTING

In this section we describe some general characteristics of multi-party collaborative computing such as DisCSP and DCOP. A DisCSP consists of a number of variables $x_1, x_2, ..., x_n$, whose values are taken from finite, discrete domains D_1, D_2, ..., D_n, and a set of constraints on their values. There are inter-agent constraints and inra-agent constraints. Solving a DisCSP is equivalent to finding an assignment of values to all variables such that all constraints are satisfied [15]. The participants exchange information related to their value assignment to the variables to check if the constraints are satisfied. Additionally in DCOPs, a global objective function is optimized. The exchange of information related to the value assignment to the variables is analogous to negotiation with each other and (re)assigning values to all variables is like reaching an agreement upon the value assignment to the variables that satisfies the constraints. For example, in the meeting scheduling problem, the collaborative agents find an agreed upon meeting date/time slot that satisfy all the constraints (e.g. do not conflict with other personal schedules) and maximizes or minimizes the global objective function (e.g. most preference, least cost) of the group.

3 PRIVACY PROTECTION MODEL

The privacy loss model of collaborative computing consists of three parties: data subject or owner, data user and third party. When part of the personal data is shared with a data user, the shared private data may get disclosed to third parties from the data user's system. Besides intruders, the data user may disclose the shared data to unauthorized parties (Figure 1).

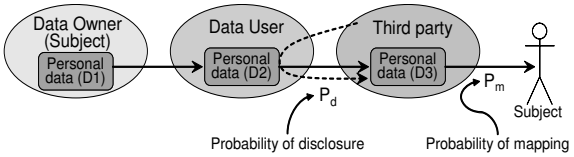


Figure 1: Privacy loss model.

It can be said that privacy loss takes place if (a) the data user can disclose the shared private data to any unauthorized entities and (b) the data receiver can map or associate the shared private data with the specific individual. The conventional distributed model is multi-agent based where the participating agents are stationary and they communicate through remote messages. In our proposed mobile agent based privacy enhancing model (Figure 2), the participating (mobile) agents along with their private data migrate into a neutral agent server platform into which they are trapped, interact locally, and share their private data to perform the desired computation. They are restricted from leaking out anything and only the computational result is sent to the users by a trusted service agent. The service agent performs security check on the computational result to protect the participating agents from hiding information in it.

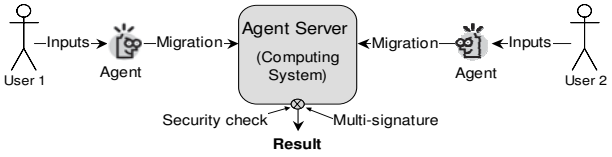


Figure 2: Proposed multi-party collaborative computing model.

Since, each participating agent uses the private data of other participating agents, for fair and uniform control over them, the agent server should be controlled by a third party (e.g. service provider). The restrictions imposed on the data users (mobile agents) should not have any impact on how the data are utilized in solving the problem. The desired control over the data users can be achieved by controlling the system resources that they use. In the proposed mobile agent based model, we refer the agent server platform as the iCOP (isolated Closed-door One-way Platform) within which the agents can interact and negotiate with one another like in the conventional distributed model and have the autonomy to implement their own strategies in solving the problem. The agent server does not restrict the agents to any specific algorithm i.e., they can use any convenient algorithm based on their goals.

3.1 iCOP Architecture

iCOP is an agent execution environment for the participating agents of multi-party computation, isolated from user hosts and user direct control over their agents. It is a closed-door platform from where the participating agents cannot communicate with the outside world. It is a one-way platform, that is the participating agents are allowed to only enter into the iCOP host platform with proper authorization but are not allowed to leave the platform. On completion of their task, all the participating agents are disposed at the iCOP host. iCOP architecture consists of two basic units: a) Management unit and b) Computational unit. Figure 3 shows a simple conceptual diagram of iCOP architecture.

The computational unit, consisting of the participating agents, performs basic computations. The input data, which it needs to solve the problem, come along with the agents through this channel. The management unit oversees the operations of the computational unit, monitors and controls the resources which the computational unit may use to perform it's computation. The management unit exercises access control over the computational input channel and output channel.

In order to make the decisions necessary to properly oversee the operation of the computational unit, the management unit uses policies that govern and constrain the behavior of the computational unit. Mandatory policies are security related

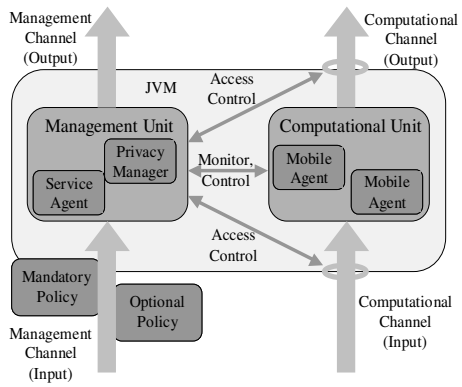


Figure 3: Logical structure of iCOP.

policies and optional policies helps in service maintenance, say by creating log files.

The privacy manager of the management unit is the controller, which enforces the specified privacy policies. It monitors the resource access request activities and based on the specified policies, it grants or denies the access to the resources for the requestor. The service agent of the management unit coordinates among participating agents in solving the problem and sends only the computational result to the users. The type of coordination and its protocol may vary depending upon the type of application. But, a service agent has four responsibilities: (1) coordination among the participating agents, (2) participate in the verification that (probably) no private data has been encoded into the computational result, (3) send the computational result, and (4) dispose the participating agents.

3.2 Service Protocol

A registered user can initiate the service by sending a request to the service agent. The initiator agent must give all of the initial parameters to the service agent before it migrates into the iCOP host. The service agent invites other participants to join the computation and provides the initial parameters to them including list of participants, problem description such as name of the problem, list of variables to be assigned, the domain of the variable values, the domain of personal valuation of certain variables etc. The initial parameters are sent to the service agent before the participating agents migrate into iCOP and before they share their private information. Thus, the initial parameters cannot contain other agents' private data and it is safe to send them out of iCOP host with the invitation. Upon getting the invitation, all participating agents collect related necessary data based on the supplied initial parameters for the computation and migrate into the iCOP host. The participating agents interact with each other and carry out the computation by sharing their personal data.

In DCOPs and DisCSPs, all of the participants solve a common (set of) problem by negotiating their value assignment to the variables and reach an agreement on the value assignment, which satisfies all the constraints [11][15]. The participating agents must follow a pre-defined format (defined by several

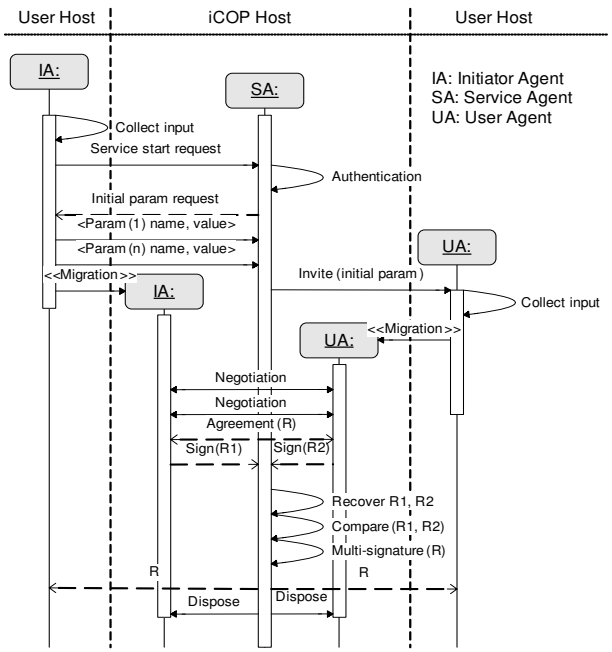


Figure 4: Service protocol sequence diagram.

properties such as letter case, variable-value pair sequence, date format, decimal point precision etc.) for creating the result consisting of set of variable names and their values. They sign the computational result individually and pass it to the service agent, which then recovers the result from individual signed message with their respective public key. The service agent verifies that the participating agents signed the same message (i.e., none of them has not encoded hidden information into the result) by exact matching the recovered messages with each other. After this verification, the service agent creates the combined multi-signature [12] and sends the signed result to the users. Finally, it disposes the participating agents at the iCOP host. Figure 4 shows the simple service protocol sequence diagram.

3.3 Result Format

The computational result consists of a set of variables and constants. The result format defines the rules of constructing a single message with its components' name-value pairs so that the result created by individual agent match with each other. Each of its component value has a data type. We broadly classify the data types as (1) integers (2) real numbers (3) date (4) string, and (5) boolean and assume that the variable names are string. The format should define at least following characteristics (where applicable).

1. Component format

- Data type of each component (integer, string, date etc.)
- Character case of string type
- Number of digits after decimal point of real numbers
- Date format ("yyyy/mm/dd", "mm/dd/yyyy" etc.)

- Boolean value (T/F or True/False)

2. Appearance order of the component_name-value pairs.
3. Separation character between component_name-value pairs.

Suppose, the result consists of four components- one constant and three variables x, y and z. Let the defined characteristics be: (1) component format- {constant}<string><uppercase>, x=<real><two digits> y=<integer> and z=<date> <yyyy/ mm/dd>, (2) appearance order- {constant}, z, y, x, and (3) separation characters- a comma followed by a space ", ". The first line of figure 5 is a valid result and the rest of the lines (2-6) are not valid according to the rules stated above. The figure points out the positions with circles where it violate at least one of the rules.

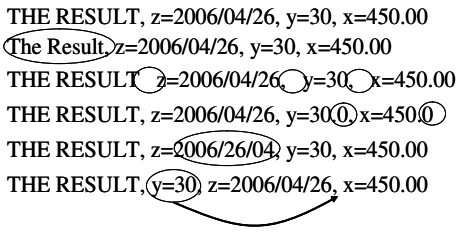


Figure 5: Good (line 1) and bad (lines 2-6) formats of the result according to the specified rules.

The rules mentioned here are not fixed for every context, but it is necessary to define those characteristics of the components to create a guideline for the participating agents to create identical results (messages) that are to be matched with each other by the service agent. In general, fixing the string case or a specific date format or the appearance order of the components does not affect the semantic much. Allowing different formats to represent the result creates the scope of hiding information into it (as shown in Section 4.2) and enforcing them to a fixed format prevents data hiding into it.

3.4 The Parallel Multi-signature Scheme

The parallel multi-signature scheme allows multiple signers to sign a message (result) separately and then combine all individual signatures into a multi-signature [5][12]. It verifies the authentication of the sender, message integrity and non-repudiation. The signature process also ensures that all of the signers sign the same message (result). We adopt the parallel multi-signature scheme presented in [12].

3.5 Implementation Issue

The Java architecture has a reference monitor, which can monitor and control the use of system resources [6]. But, the Java security manager enabled JVM itself is not a one-way closed-door platform. Thus, we use the Java architecture and customize the Java security manager to make the platform one-way and closed-door. The Java technology can provide security to end user system against untrusted code (e.g., applet from the Internet) by putting them into sandbox and protecting them from security sensitive activities on local system. However, it allows downloaded code in the sandbox to connect back to the originator i.e., there exists open channel between the sandbox of end user host and the originator host of the downloaded code. So, if the downloaded code can get some information (due to sloppy security policy) from local system, it can send the information to its originator. In iCOP, even the server can be made secured from mobile agents by activating the Java security manager with proper policy, the agents of multi-party computation exchange their private information in their problem solving process, which is unavoidable. Thus, the agents get sensitive information at the first place (due to the nature of the multi-party collaborative problem) without compromising the security manager and can send the acquired information to their originator hosts.

In iCOP, for any system resource access request, the privacy manager inspects the system class stack corresponding to the current series of method call and checks if there is any external class (i.e. migrated agent) in the system class stack by checking their code bases. It denies the access request if it finds any external class in the class stack. This restricts the mobile agents from any type of communication with the outside world or from migration to other hosts from the iCOP host.

Customizing the Java security manager to close all channels, which may be used by the mobile agents, can prevent them from leaking out any information. However, without sending the computational result to the users, the system becomes useless. It needs a mechanism to send the computational result to the users. iCOP uses a trusted service agent for this. To ensure that the mobile agents don't send hidden information through the result sending process, the service agent (along with the participating agents) performs security check and creates a multi-signature on the computational result before sending it to the users.

A service agent can be a signed agent from a reputed developer and installed locally on iCOP which is capable of carrying out four jobs as listed in Section 3.1. It is assumed to be trusted and not malicious. Many service instances can be run by instantiating separate Java virtual machine for them. A user may build her own participating mobile agent or may use agents created by professionals.

4 ANALYSIS

Data disclosure requires a disclosure channel (open or covert) from the agent platform to the outside world. In iCOP, the participating agents are protected from accessing system resources, using which they can leak out the personal data acquired in the negotiation with other agents or take them away during migration to other hosts. So, the participating agents cannot use an open channel. However, they may try to leak information though a covert channel.

4.1 Covert Channel

Covert channels generally remain unnoticed because of covert means in which information is communicated. From various covert channels explored by researcher [1][13], we can say that there are at least three basic requirements for a covert channel. (1) The sending and receiving process must have access to the same attribute of the shared resource (variable). (2) The sending process must be capable of changing the attribute. (3) The receive process must be capable of detecting the change in attribute. Figure 6 shows the modular concept of covert channel. There are two types of covert channels [1] (a) storage channel and (b) timing channel. Various examples can be found in papers [1][13].

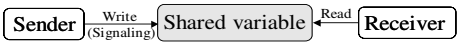


Figure 6: Covert channel.

All processes including the participating agents use some common system resources like memory and CPU. But, other system resources like file system, communication socket, printer etc. are restricted for the participating agents. However, since a large number of processes share those common system resources, covert channels through those resources are extremely noisy and the channel bandwidth will be very low. Besides, the receiving process must reside in the system itself to probe the attributes of those common resources to interpret the sent data.

The computational result is shared between the participating agents and the outside receivers (i.e. users). Thus, it is a potential noiseless covert channel between a participating agent and a result receiver. A participating agent may try to encode the acquired private data of other agents into the computational result and try to send the innocent-looking computational result to its user through the service agent.

The art of sending hidden data is known as steganography and the analysis of steganography to detect hidden message is called steganalysis [2]. Following we present some steganography techniques to analyze their characteristics.

4.2 Steganography and Steganalysis

Steganography generally requires some kind of modulation on the original object. In our context, the original object refers to the computational result created by the agents based upon their agreement in the negotiation using the pre-defined format. The modulation technique uses some kind of protocol to encode information and the receiver need to perform related demodulation to interpret the encoded data. Different kinds of modulation are possible. Following are some examples.

Adding text: The sender adds additional characters like whitespace, punctuation marks or decimal point in numerical values etc. without changing the semantic of the text. Table 1 shows a simple protocol.

Suppose, the first line of Figure 7 is the original object; then with the protocol shown in Table 1, the second line contains hidden data “010000011110”. But, semantically both lines are equivalent.

Table 1: Steganography protocol by adding characters

Signal	Means
Additional 1 whitespace	00
Additional 2 whitespace	01
Additional 1 digit with numerical value	10
Additional 2 digit with numerical value	11

This is the computational result, x=3, y=2
This␣is␣the␣computational␣result, x=3(00), y=2(0)

Figure 7: Hiding data by adding extra characters.

Arranging components: An object may possess the same semantic even after re-arranging its distinguishable components. For example, a date may be represented with different formats as shown in table 2. With the protocol shown in Table 2, to send “101” the sender must send a date value in the “yyyy/dd/mm” format. Similarly, by choosing a sequence of the variable_name-value pairs, a number of bits can be transferred covertly.

Table 2: Steganography protocol by arranging components

Format	Means	Format	Means
dd/mm/yyyy	000	mm/yyyy/dd	011
dd/yyyy/mm	001	yyyy/mm/dd	100
mm/dd/yyyy	010	yyyy/dd/mm	101

Changing case: Data can be encoded by changing the case of certain alphabets (e.g., start of each sentence, first alphabet of each word or an alphabet of any position) of the text. With the protocol shown in table 3, the text “This is the Computational Result” contains hidden data “10011”.

Preventive Mechanism: In order to prevent agents from hiding private information in the computational result, iCOP policy obligates the participating agents to represent the computational result individually in a pre-determined format defined by several characteristics of the result components as described in Section 3.3. This policy is enforced by the requirements (which is checked by the service agent) that the computational results passed by individual agent must match with those of other agents. The participating agents mutually protect each other from encoding data into the result by passing its copy of the result to the service agent. Note that, it is necessary for the participating agents to have a common computational result (which is a characteristic of multi-party collaborative computing as described in Section 2) created by negotiation and known to each of them. Thus, if a participating agent encodes secret information in its computational result, the result passed by it to the service agent will not match

Table 3: Steganography protocols by changing case of alphabets

Format	Means
Capital letter	1
Small letter	0

with those passed by all other participants regardless of the encoding method and the content type of the computational result.

The service agent can only detect that some of the participants probably has encoded secret information in its result (when there is any mismatch) but cannot determine which agent(s) has done that. When detected hidden data in the result, it will not send it to any of the users. Thus, we see that the identical result requirement policy protects participating agents from using covert channel through the result sending process.

Non-modulating: Information may also be sent with the computational result without encoding into the computational result. In our investigation, we found one such approach that we call the result biasing method. If there are s numbers of possible solutions of a given problem, an agent may try to bias all participants towards a specific valid solution in the negotiation process. For example, in the meeting scheduling problem, if there are 4 slots (among a number of candidate slots), in any of which all of the participants can attend the proposed meeting (i.e., all of the constraints are satisfied), the bits '00', '01', '10' and '11' may be signaled to the receiver by biasing the meeting slot into solution number 1, 2, 3 and 4 respectively. The number of transferable bits by this technique is,

$$b = log_2(s) \quad (1)$$

The encoding/decoding protocol between a sender inside iCOP and an outside receiver must be defined before the agent migrates into iCOP. Thus, the protocol will depend upon the knowledge of the agent about the possible solution set of the problem.

Table 4: A simple example of result biasing protocols

Soln. no.	Means	Soln. no.	Means
1	000	2	001
3	010	4	011
5	100	6	101
7	110	8	111

In conclusion, we can say that in iCOP, the open channels as well as covert channels using system resources are protected by the privacy manager. The potential covert channel through the result sending process is checked by the trusted service agent in cooperation with the participating agents. But, the result biasing method for sending hidden information may not be possible to detect, avoid, or prevent. However, it is a transient channel [1] through which only few bits may be possible to leak out.

5 EVALUATION

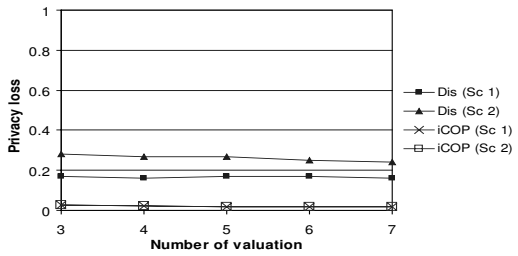
The main purpose of the proposed mobile agent based computing model and its required components is to enhance privacy in multi-party collaborative computing. Thus we evaluate the model by comparing the privacy loss for a collaborative problem with the well-known ADOPT algorithm [11] in this model and in the traditional distributed model. Since, the agents solve the problem by using a server, instead of multiple hosts like in distributed model, we also compare the computational costs in the two systems.

5.1 Privacy

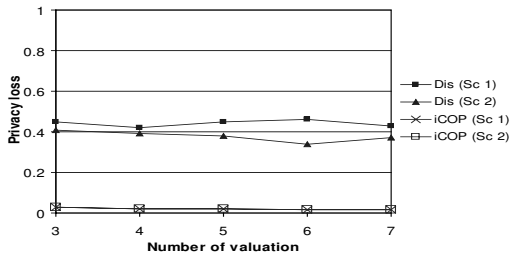
The meeting scheduling problem is a multi-party collaborative computing problem where privacy might be important. Thus, to evaluate the effectiveness of the proposed mobile agent based model in privacy protection, we consider the meeting scheduling problem. Researchers commonly observe that the majority of scheduling instances will consist of a small number of meetings that need to be negotiated simultaneously [7][16].

In our investigation, we found that the privacy loss in iCOP may occur through a covert channel using the result biasing method, which we described in Section 4.2. This channel is a transient channel [1] that can transfer a fixed number of bits, which depends upon the number of valid solutions of the given problem that meet all the constraints. Privacy loss was measured using VPS metric [9], which accounts for the fraction of participants to whom the private information (represented in states) is revealed and the fraction of states, which are revealed to others. Privacy loss was also measured in MAX metric [7], which accounts for the maximum private information revealed to any participant. We considered the ADOPT algorithm for two scenarios in conventional distributed model and in our mobile agent based model. For the purpose of analysis, maximum privacy loss in iCOP is shown. Figure 8(a) shows privacy loss in VPS (EntropyTS) metric for different number of valuation for two scenarios. Figure 8(b) shows the graph in MAX metric for the same settings. The maximum privacy loss in iCOP is limited by the number of transferable bits through the result sending channel and can be said to be independent of algorithm used to solve the problem (unless the algorithm itself results in less privacy loss).

The privacy loss in iCOP decreases with the number of valuations because the measurement counts the fraction of states that have been revealed. In iCOP, a fixed number of bits are revealed making the fraction decrease as the total states increase. ADOPT algorith was shown to be the most efficient (using conventional distributed computing model) among the available DCOP algorithms in privacy protection by algorithmic mechanism [7]. The amount of privacy loss in other algorithms are higher than that in ADOPT algorithm. Thus, we can conclude that the available distributed algorithms can en-



a) Privacy loss in VPS(EntropyTS) metric



b) Privacy loss in MAX metric

Figure 8: Privacy loss in ADOPT algorithm using distributed model and iCOP.

hance privacy protection by using our proposed mobile agent based computing model.

We believe that leaking information through a covert channel using result biasing method might be very difficult because it needs to have a number of valid solutions and also the sender needs to bias all participants to a specific valid solution.

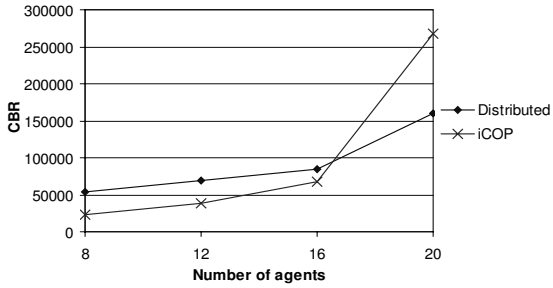
5.2 Computational Time

For computational time measurement, Cycle-Based Runtime (CBR) metric [3] appears to be the most appropriate one for distributed algorithms. It considers concurrency of the agents and latency of the underlying communication environment as described in paper [3]. The computational unit of distributed algorithm is expressed with cycle, which is defined as the time duration in which all agents receive all their incoming messages and send out all their outgoing messages [11]. Total runtime of m cycles is represented by

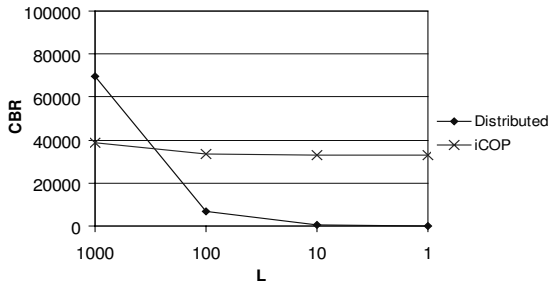
$$CBR(m) = t \times ccc(m) + L \times m \qquad (2)$$

where, t is the time required for one constraint check, m is the number of cycles, $ccc(m)$ is the number of concurrent constraint check in m cycles and L is the communication latency between cycles. The value of L is represented in terms of number of constraint check, considering that one constraint check is the smallest unit of time. For measuring the communication latency, we used two hosts (Host A and Host B) connected in typical campus network. Host A was an IBM Laptop running Microsoft Windows XP with 768 MB of RAM, 100Mbps LAN card, 1.4 GHz Intel processor using the SUN Java 2 Software Development Kit (J2SDK) 5.0. Host B was an IBM Laptop running Microsoft Windows XP with 256 MB of RAM, 100Mbps LAN card, 1.8 GHz Intel processor using the SUN Java 2 Software Development Kit (J2SDK) 5.0. In host A, a total of 10^6 language level key instruction took about 8ms and round-trip messaging latency between the two hosts was measured as about 8ms. Considering no more than 1000 language level key instructions per constraint check, the value of L for message exchange in the distributed model will be no less than 1000.

On the other hand in iCOP, the latency for local messages between two agents is very small and can be assumed to be no greater than one constraint check. But all of the participating agents need to migrate to the agent server, increasing the latency in iCOP. For a 23KB agent, it took about 21ms to migrate (including serialization and de-serialization) from one host to another in Aglet environment [8]. So, we consider that the latency for agent migration is no more than 3000 in our environment. It is noteworthy that unlike distributed model, the execution cycle in iCOP is serial instead of concurrent. Also note that the values that we found in our measurement will not be the same in other environments.



a) For different number of variables



b) For different values of L

Figure 9: Comparison of computational time for ADOPT algorithm in distributed model and in iCOP using CBR.

Figure 9 shows the CBR for the graph 3-coloring problem (which we take as an example to show the effect of large number of variables) with ADOPT algorithm in the mobile agent based model and in conventional distributed model for the same experimental settings. From Figure 9(a) we see that iCOP outperforms distributed model for small number of variables. The CBR of ADOPT in the conventional distributed model is high mainly because of its large number of cycles, which increases the communication latency. But for higher number of variables, the total number of (serial) constraint check in iCOP becomes very large. Thus, we can conclude that when the number of variables is small, ADOPT

takes less cycle-based runtime in our mobile agent based model than in the distributed model. In the networks that are faster than the one we used in our experiment, the performance of distributed model increases significantly. Figure 9 (b) shows (for 12 agents) that the distributed model outperforms iCOP in high speed networks having lower value of latency between cycles.

The computational time of another well-known algorithm OptAPO [10] is expected to be worse than that of ADOPT [3]. Thus, we believe that for small number of agents, the computational time of OptAPO in our mobile agent based model will also be less than that in the conventional distributed model, when used in typical networks. But, in very high speed networks and for large number of agents, the conventional distributed model will outperform mobile agent based model, since the constraint check are effectively serial in mobile agent based model that uses agent server.

6 CONCLUSION

Privacy is viewed as a crucial issue in autonomous systems, like in ubiquitous environment. We have presented a mobile agent based privacy enhancing computing model for multi-party collaborative computing. We evaluated the proposed model in terms of privacy loss and found that the privacy protection using our proposed model is very efficient and probably only few bits of data can be leaked out through covert channels. In a very high speed network, our proposed model requires larger computational time than the conventional distributed model, when the number of participants is very large. However, for smaller number of participants, the proposed model requires less computational time than the distributed model. While considering computational time, the proposed model is suitable for typical networks that are available widely and especially more suitable for slower networks like wireless networks.

REFERENCES

[1] American National Computer Security Center, A guide to understanding covert channel analysis of trusted systems, http://www.fas.org/irp/nsa/rainbow/tg030.htm, NCSC-TG-030, Ver. 1, (1993)

[2] Bennett, K., Linguistic steganography: survey, analysis, and robustness Concerns for hiding information in text, CERIAS Tech Report 2004-13, (2004).

[3] Davin, J., Modi, P., Impact of problem centralization in distributed constraint optimization algorithms, Proc. of Autonomous Agents and Multi-Agent Systems (AAMAS'05), pp. 1057-1063, (2005).

[4] Du, W. and Atallah, M. J., Privacy-Preserving Cooperative Scientific Computations, Proc. of the 14th IEEE Workshop on Computer Security Foundations, Canada, pp. 273-282 (2001).

[5] Frankel, Y. and Y. G. Desmedt, Parallel Reliable Threshold Multisignature, Tech. Report TR- 92-04-02, Dept. of EE. and CS., Univ of Wisconsin - Milwaukee (1992)

[6] Gong, L. Ellison, G. Dageforde, M., Inside Java 2 Platform Security: Architecture, API Design, and Implementation (2nd edition), Addison-Wesley Professional (2003).

[7] Greenstadt, R. Pearce, J. P. Bowring, E. and Tambe, M., Experimental analysis of privacy loss in DCOP algorithms, Pro. of Third Int. Joint Conf. on Autonomous Agents and Multiagent Systems (AAMAS'06), Hakodate, Japan, (2006).

[8] Lange, D.B. Oshima, M., Programming and Deploying Java Mobile Agents with Aglets, Addison-Wesley, (1998).

[9] 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, Journal of Autonomous Agents and Multi-Agent Systems, Springer, Vol. 13, No.1, pp. 27-60 (2006).

[10] Mailler, R. and Lesser, V., Solving distributed constraint optimization problems using cooperative mediation. Pro. of Third Int. Joint Conf. on Autonomous Agents and Multiagent Systems (AAMAS'04). New York, pp. 438-445 (2004).

[11] Modi, P. J. Shen, W. Tambe, M. Yokoo, M., ADOPT: Asynchronous distributed constraint optimization with quality guarantees. Artificial Intelligence Journal (AIJ). Vol. 161, pp. 149-180 (2005).

[12] Shieh, S.P. Lin, C.T Yang, W.B and Sun, H.M., Digital Multisignature Schemes for Authenticating Delegates in Mobile Code Systems, IEEE Transaction on Vehicular Technology, Vol. 49, No. 4, pp. 1464-1473 (2000).

[13] Tsai C. R. and Gligor, V. D., A Bandwidth Computation Model for Covert Storage Channels and Its Applications, IEEE Symposium on Security and Privacy, pp. 108-121, (1988).

[14] Yao, A., Protocols for secure computations, Proc. of the 23rd Annual IEEE Symposium on Foundations of Computer Science, (1982).

[15] Yokoo, M. Durfee, E. H. Ishida, T. and Kuwabara, K., The Distributed constraint satisfaction problem: formalization and algorithms, IEEE Transactions on Knowledge and Data Eng., Vol.10, No.5, pp. 673-685 (1998).

[16] Yokoo, M. Suzuki, K. and Hirayama, K., Secure distributed constraint satisfaction: reaching agreement without revealing private information, Artificial Intelligence, Vol.161, Issue 1-2, pp. 229-245, (2005).