Privacy Protection with Customized Java Sandbox Architecture

Md. Nurul HUDA† Eiji KAMIOKA‡ and Shigeki YAMADA‡

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

Abstract Multi-party cooperative problem solving techniques require multiple inputs from multiple participants. The participants may want to keep their inputs secret from one another. However in traditional agent based systems, the inputs are exchanged directly with one another and can be disclosed to their users resulting in high privacy loss. We use a customized Java Sandbox architecture for an agent server in which the agents exchange their private inputs, compute the result and give it to a trusted service agent. The agents are protected from communicating with the outside world and from leaving the agent server. The service agent checks for probable private data in the computation result, creates multi-signature on the result and send it back to the users. Participating agents are disposed after the completion of the job. We also analyze probable covert channel through the result sending process. The presented architecture can be very effective in protecting privacy in multi-party cooperative computing.

Keyword Privacy protection, mobile agent, multi-party cooperative computing, covert channel, steganography

1. Introduction

Multi-party cooperative computing, such as the distributed constraint satisfaction problem (DCSP), distributed constraint optimization problem (DCOP) [1,2] solving, involves multiple inputs from multiple participants. The participants solve a common problem based on the inputs they each supply. In many contexts, such as the meeting scheduling, the participants may want to keep their inputs secret from one another. Traditionally DCSP are dealt with software agents, which act and take decisions on behalf of their users. One reason for solving a DCSP in a distributed fashion is that agents might not want to communicate all the information to the centralized leader agent [1]. In the centralized approach, the privacy loss of the participants to the central agent or the leader agent is high. However, Depending upon the metrics used for privacy loss measurement, the relative privacy loss in an approach might vary with respect to other approaches [3, 4]. The main technique used for reducing privacy loss in a distributed algorithm is to reduce the amount of shared information with others. However, all these algorithms have some inherent privacy loss because the participants need to share their private inputs to resolve the conflicts among them or to optimize the solution.

Another method of reducing privacy loss is to remove or hide the relationship of the personal data with the identifiable individual [5,6]. This is commonly referred to as anonymization. In a multi-party computation and for a known number of participants, the relationship of the data with an individual is bounded by the number of individual (*k*) i.e., no less than (*1/k*). This approach is more suitable in the cases where the number of data owners is very large, such as in privacy preserving data mining where there are a large number of individuals related to the data set.

Besides non-cryptographic approaches, cryptographic approach for protecting privacy has also been proposed [7,8]. These approaches incur a large overhead, theoretical and not suitable for practical applications.

In traditional multi-party computing models, the data owners or their agents retain no control over the data that they give to others or on the data users (agents) that use the given data. A data user may utilize the given private data for an unintended job or disclose the given data to an unauthorized third party resulting in privacy loss.

In this paper, we propose a new mechanism for reducing privacy loss in multi-party collaborative computing systems. In our proposed computing model, mobile agents are provided user inputs which then migrate into an agent server provided by a neutral service provider. We describe an agent server platform architecture named iCOP (isolated Closed-door One-way Platform) into which the participating mobile agents are trapped. To solve the problem, they negotiate and interact with one another by sharing their private input data like the distributed model. The privacy manager of the iCOP architecture restricts the participating agents from disclosing the acquired input data of other participants to the outside world. It also restricts the participating agents from leaving the platform with the learnt private data. A service agent checks the computational result for probable private data hidden with the computational result, creates a multi-signature along with the participating agents and sends the result to the users. In order to ensure protection, finally the participating agents along with the learnt data are disposed at the iCOP server.

2. Privacy loss model

When part of the personal data is shared with the data user, the shared private data may get disclosed to third parties from the data user's system (Figure 1). Besides intruders, the data user may disclose the shared data to unauthorized parties. But acquiring data by an unauthorized third party does not necessarily mean privacy loss. If the third party cannot map or associate the acquired data with the specific individual we do not say it as privacy loss.

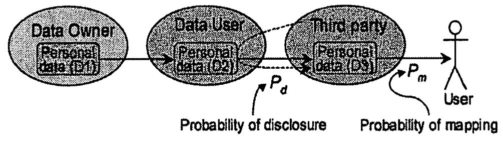

Fig 1: Privacy loss model

Thus we can define privacy loss from the data user's system as,

$$\psi = P_d \times P_m \times i \tag{1}$$

where, P_d is the probability of disclosure from the data user's system, P_m is the probability of associating or mapping the acquired data with the identifiable individual by the unauthorized party and i is the size of the shared data.

Let the ideal privacy level of a user be one and the worst-case privacy level be zero. With this normalization the privacy loss becomes,

$$\psi = \alpha \times P_d \times P_m \times i \tag{2}$$

where, I is the personal data used for the computation and $\alpha = 1/I$.

In conventional multi-agent based systems, (a) the agents are free to disclose the shared private data to their users (i.e., P_d = 1) and (b) the receiving agent can identify the sender agent and thus can map or associate the received private data with the sender agent/user (i.e., P_m =1).

Among different approaches to reduce the privacy loss, anonymization technique and some algorithms reduce the value of P_m and some distributed algorithms reduces the value of i. In this research our aim is to reduce privacy loss by reducing the value P_d.

3. Privacy Protection Model

In an ideal privacy preserving multi-party cooperative computing model, the computing system should take the inputs from the participants and give back the result keeping the inputs secret from each other.

Our mechanism of privacy protection is to provide inputs to mobile agents, bring them into a neutral agent server, allow them to interact locally to share their private data and perform the desired computation, restrict them from leaking out anything and send only the computational result to the users. Since each agent uses the private data of other agents, the agent server should be controlled by a third party. In our proposed computing model the data users (i.e., mobile agents) are trapped into an agent server and restricted in a way such that they cannot disclose the shared private data to unauthorized parties (Figure 2).

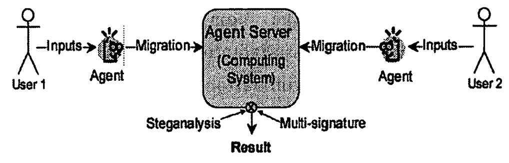

Fig 2: Proposed multi-party cooperative computing model

The service provider should impose neutral and uniform control over all the data users (mobile agents) and it 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 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 the conventional distributed approaches and have the autonomy to implement their own strategies in solving the problem. The agent server does not restrict the agents to any specific approach- centralized or distributed i.e., they can use any convenient algorithm based on their goals.

3.1. iCOP Architecture

iCOP is an agent execution environment for 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 participating agents of the multi-party computation constitute the computational unit and perform basic computations. The management unit oversees the operations of the computational unit, monitors and controls the resources that it may use to perform it's computation and so on. The management unit exercises access control over these computational input channel and output channel.

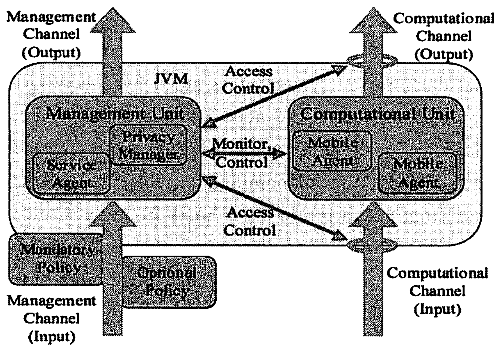

Fig 3: Logical structure of iCOP model

In order to make the decisions necessary to properly oversee the operation of the computation of the computational unit and to achieve flexibility, management unit will carry with them, or otherwise have access to, *policies* that govern and constrain their behaviors, current mission and strategy.

The *privacy manager* of the management unit is the controller that enforces the specified policies. It monitors the resource access request activities and based on the specified policies, it grants or denies the access to the requestor. The closed-door and one-way properties of the iCOP host platform are achieved by trapping any direct or indirect access requests for communication resources by the participating agents and denying their requests.

A *service agent* of the management unit coordinates among participating agents in solving the problem and sends the computational result to the users. The type of coordination and its protocol vary depending upon the type of application. Thus different application needs different service agent. But all of them have four common responsibilities: 1. coordination among the participating agents, 2. verification of the computational result that no private data has been encoded into it, 3. send the computational result, and 4. dispose the participating agents.

The coordination of the service agent includes distributing the list of participants, inviting them to join the computation and passing them initial parameters of the problem. The initial parameters includes 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. Based on the initial parameters, the participating agents decide which data to bring into the iCOP host for the computation. The service agent also performs steganalysis on the computational result created by the participants to detect any hidden private data in the computational result.

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 so that they cannot contain other agents' (shared) private data (that will be shared inside iCOP later) and are safe to send out from the iCOP host with the invitation. The service agent invites other participants to join the computation and provides the initial parameters to them. 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. Finally, they reach an agreement and create the computational result.

In DCOP and DCSP, such as the meeting scheduling problem, all the participants solve a common (set of) problem by negotiating/exchanging their value assignment to the variables and reach an agreement that satisfy all the constraints [1,2]. In other multi-party applications, such as the buying-selling application, two or more participants create an agreement or contract with one another as the computational result.

In the parallel multi-signature process, the participating agents sign the computational result individually and pass it to the service agent, which then recovers them from individual signed message with their respective public key.

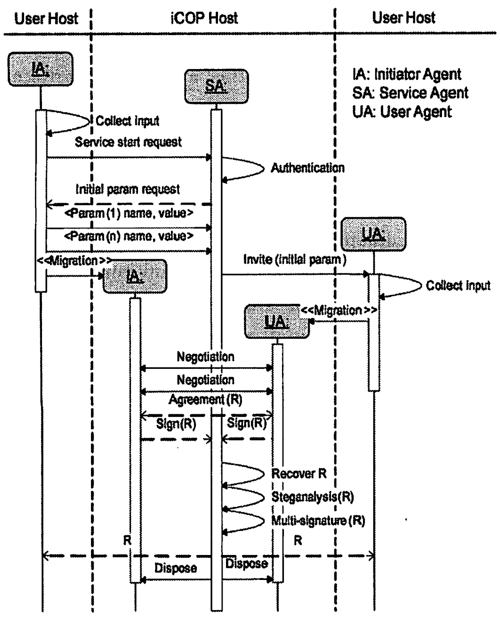

Fig 4: Service protocol sequence diagram

The group of agents who create the result must follow a pre-defined format for creating the result consisting of set of variable names and their values. The service agent checks the compatibility of the recovered computational result with the pre-defined format. The service agent verifies that the participants have 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 (which also verifies that all of the participants has signed exactly the same message) and sends the signed result to the users. Finally it disposes the participating agents at the iCOP host.

3.3. The Parallel Multi-signature Scheme

The parallel multi-signature scheme allows multiple signers to sign a message separately and then combine all individual signatures into a multi-signature. 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. We adopt the parallel multi-signature scheme presented by [9]. Because of space limitation we skip its description.

3.4. Implementation issue

The main objective of the iCOP architecture is to prevent participating agents from disclosing any data (other than the computational result) to the outside world. Data transfer from a participating agent in the iCOP can occur in many different ways. For example, direct transfer through message, mail, RMI etc. Even data may be encapsulated in an agent and transferred by transferring the agent itself. However, all these data transfer techniques generally need to use network resources. Thus, data transfer can be controlled by controlling the network resources.

The Java security manager is a well-known architecture for controlling access to system resources [10]. The sandbox model can restrict external codes (external mobile agents) from using system resources to transfer data. However, it allows external codes to connect back to the originator, which should not be allowed in iCOP architecture to achieve its goals. So in iCOP, for any network resource access request, the privacy manager, which is a customized Java security manager, inspects the system class stack corresponding to the current series of method call. The privacy manager checks if there is any external class in the system class stack by checking their code bases and denies the access request if it finds any external class in the class stack. This restricts the external agents from any type of communication with the outside world or from migration to other hosts from the iCOP.

A service agent is locally installed agent specialized for specific kind of service and developed by an experienced and reputed developer. It is given privileges to dispose external agents and to communicate with the outside world so that it can invite the participants and send the result to the users. The flexibility is achieved by implementing security policies, which allow granting different privileges to different code from different codesources. A user agent is a mobile agent, possibly developed by the users or other organizations that normally reside at the user host.

4. Analysis

Data disclosure requires a disclosure channel from the iCOP to the outside world. The participating agents are protected from accessing system resources using which they can leak out the acquired personal data of other agents or take away during migration to other hosts. However a participating agent may leak information though a *covert channel* [11,12]. A covert channel requires a shared variable between the sender and the receiver. The sender puts data or some kind of signaling on the shared variable and the receiver reads the shared variable. For example, the sender may signal the receiver by changing the frequency of use of a system resource (e.g. file) which the receiver can monitor and decode the secret data from the frequency of use of the resource by the sender. Figure 5 shows the modular concept of covert channel.

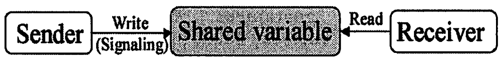

Fig 5: Covert channel

The only shared variable between a potential sender and an outside receiver in iCOP model is the computational result. Thus the result sending channel might be a potential covert channel.

The art of sending hidden message is known as *steganography* [13]. The analysis of steganography to detect hidden message is known as *steganalysis*. Assuming that the computational result is a plain text object, our analysis is limited to text steganalysis.

Text steganography generally requires some kind of modulation on the original text. The sender modulates the original text using some protocol to include hidden information and the receiver must perform related demodulation to recover the hidden text. 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.

Table 1: Simple text steganography protocols

Signal	Means
1 whitespace	0
2 whitespace	1

Signal	Means
1 decimal point	0
2 decimal point	1

With the protocol shown in Table 1, the first line of Figure 6 is the original text and the second line contains hidden data "1011". Similarly 3rd line is the original text and 4th line includes hidden data "10". But semantically they are equivalent. The number of transferable bits varies based upon the steganography protocol.

This is the computational result

This is the computational result

X=3, Y=2

X=3.00, Y=2.0

Fig 6: Examples of text steganography

Arranging components: A text may have some distinguishable components and rearranging those components may still possess the same semantic. For example, the same date "dd/mm/yyyy" may be represented with different formats as shown in table 2. If there are *n* distinct elements in the original text, then by arranging them in different ways *factorial(n)* unique values can be represented. So, the number of bits that can be transferred by choosing any one of them is

$$b = \log_2(n!) \qquad (3)$$

Table 2: Text steganography protocols with arranging components

Format	Means
dd/mm/yyyy	000
dd/yyyy/mm	001
mm/dd/yyyy	010

Format	Means
mm/yyyy/dd	011
yyyy/mm/dd	100
yyyy/dd/mm	101

With the protocol shown in Table 2, to send "010" the sender must send a date in the "mm/dd/yyyy" format.

Changing graphical properties: Data encoding is also possible by changing the case of certain characters (e.g., first character of each word) of the text.

Table 3: Text steganography protocols by changing case

Format	Means
Capital letter	1
Small letter	0

With the protocol shown in table 3, the text "This is the Computational Result" contains hidden data "10011". The number of transferable bits varies based upon the steganography protocol.

Non-modulating: Information can also be sent with non-modulating steganography. In our investigation we found one such approach that we call *result biasing*. If there are *s* numbers of valid solutions of the problem, an agent may bias the computational result towards a specific valid solution by rejecting other solutions in the negotiation process. For example, if there are 4 valid solutions of the problem, the bits '00', '01', '10' and '11' may be signaled to the receiver by biasing the result into solutions number 1, 2, 3 and 4 respectively. Thus the number of transferable bits by this technique is,

$$b = \log_2(s) \qquad (4)$$

This type of steganography can be mounted in any architecture and the same number of bits can be transferred by this attack irrespective of the architecture. Figure 7 shows the minimum privacy loss in optAPO [14] algorithm in VPS privacy metric [3] for traditional multi-agent based distributed model and iCOP model (with 8 valid solutions and 16 preference values).

Note that, all of the text steganography by text modulation change the original text. Text steganography by text modulation is detectable in iCOP with known cover analysis [13] by comparing the computational results from different agents. When they all matches with each other, all of the participants created the computational result according to their agreement in the negotiation and they followed the pre-determined format. With the predetermined format "all caps, coma separated variable_name=variable_value, two precision numerical value, date_format dd/mm/yyyy" a possible computational result that follows the format will be "DATE=23/05/2006,ITEM=CD,QUANTITY=10,PRICE =450.00". If at least one of the participants follows the correct format, then the service agent can detect any mismatch among the computational results signed by the participants.

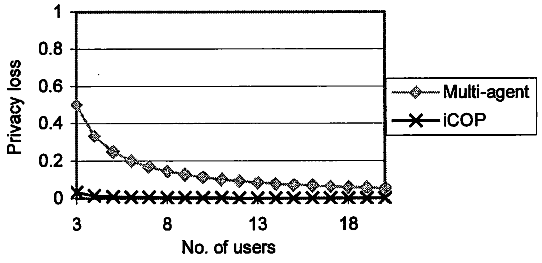

Fig 7 : Privacy loss in the meeting scheduling problem in the optAPO algorithm

The service agent cannot detect steganography that uses non-modulation technique. In fact, leakage through this technique might be successful in any architecture. However, this method might be difficult because it needs to have a number of valid solutions, need to bias all participants to a specific valid result and finally the number of transferable bit is very small. Also the sender agent may incur penalty as a group member if it tries to bias the computational result to a specific solution because that solution might not be the best solution for the group. The result sending channel is a transient channel [12], which transfer a fixed amount of data and then cease to exist.

5. Conclusion

We have presented a privacy protection model with a customized Java sandbox architecture for agent based multi-party cooperative computation and analyzed the privacy protection aspect of the model. In our analysis, we found that the privacy protection using our iCOP model is very efficient and probably only few bits of data can be leaked through covert channels. The potential application domains of our proposed model can be privacy preserving DCSP, DCOP, agent based business applications etc.

References:

[1] 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*, 10(5), 1998, pp: 673-685

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

[3] 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, Jul 2006, pp:27 – 60

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

[5] David E. Bakken, Data obfuscation: anonymity and desensitization of usable data sets, *IEEE Security & Privacy*, Vol. 2, no. 6, 2004. pp: 34-41

[6] L. Sweeney, K-Anonymity: A Model for Protecting Privacy, *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, Vol. 10, no. 5, 2002. pp:557-570

[7] Yao, A. Protocols for secure computations, *In Proc. of the 23rd Annual IEEE Symposium on Foundations of Computer Science*, 1982

[8] Du, W. and Atallah, M. J. Privacy-Preserving Cooperative Scientific Computations, *In proc. of the 14th IEEE Workshop on Computer Security Foundations*, Canada 2001, pp: 273 – 282

[9] Shieh, S. P. Lin, C. T. Yang, W. B. and Sun, H. M. Digital multisignature schemes for authenticating delegates in mobile code systems, *IEEE Trans. Vehicular Technology.* vol.49, Jul 2000, pp: 1464–1473

[10] Gong, L. Ellison, G. Dageforde, M. Inside Java 2 Platform Security: Architecture, API Design, and Implementation (2^{nd} edition), *Addison-Wesley Professional,* May 27, 2003

[11] U.S. Department of Defense. Trusted Computer System Evaluation, *Publication DoD 5200.28-STD. Washington: GPO 1985.* http://www.radium.ncsc.mil/tpep/library/rainbow/5200.28-STD.html

[12] National Computer Security center, A guide to understanding covert channel analysis of trusted systems, *NCSC-TG-030, Version-1, Nov 1993.* http://www.radium.ncsc.mil/tpep/library/rainbow/NCSC-TG-030.html

[13] Krista Bennett, Linguistic steganography: survey, analysis, and robustness Concerns for hiding information in text, *CERIAS Tech Report 2004-13*, https://www.cerias.purdue.edu/tools_and_resources/bibtex_archive/archive/2004-13.pdf

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