Introducing Robust and Private Computation into Grid Technology

Regine Endsuleit and Jacques Calmet
Universität Karlsruhe (TH), Germany
E-mail: {endsuleit, calmet}@ira.uka.de

Abstract

We propose to adapt the recently published model for secure multi-agent computation from [4] to Grid technology. Since the model is based on secure multi-party computation we achieve computations that are robust against a (user-defined) maximum number of wrong or missing inputs. Furthermore, all computations and data stay private until they are returned to their originator. Being able to use mobile agents in a per se non-trusted environment like a Grid opens up a variety of possibilities for sensitive applications.

Key Words: Security in the Grid, Distributed Computation, Robustness, Privacy, Secret Sharing

1. Introduction

When the term ”the Grid” has been introduced in the mid1990’s in [7] it was mainly meant to be a powerful technology providing scientists and engineers with an infrastructure for distributed computations (for a description of the architecture see [6]). Since then a lot of research has been done and there are dozens of major Grid projects under development in scientific and technical computing, research and education. For example NASA’s *Information Power Grid*, the Dutch *Distributed ASCI Supercomputer* and *Tera Grid* in the USA that links major academic sites. However, there is a considerable consensus on key concepts and technologies. This consensus is largely represented by the open source Globus Toolkit$^{\text{TM}}$ [8] that is used by each of the projects.

We are at a stage where industrial interest is becoming obvious while convergences with E-science and E-business have been well identified. On the security side, Globus Toolkit offers reliable solutions to authentication, authorisation and resource discovery and monitoring. At the same time, there are deficiencies on the protocol dealing with error propagation or with heterogeneous bases just to name two of the present weaknesses. From the security point of view, there are also some significant functionalities missing. Among them is the virtualisation of hosting environments. Finally, not enough progresses have been reported on dependability and end-to-end QoS. If you adopt the Globus model as an illustrative framework, we must incorporate our security mechanism within the OGSA service model which is responsible for providing a reliable secure management of distributed states. This implies that all services adhere to specified Grid service interfaces and behaviours dealing in particular with management, authorisation and notification. Few of the services are persistent, most are transient services. This means that when considering an application we introduce a new service and some security mechanisms for this service. Such services lead to define concepts of Factory, Create Operation, Grid Service Handle (GSH), Grid Service Reference (GSR) and a Mapper interfacing a GSH to a GSR. An approach to authentication is that protocol binding handles authentication during the invocation of any Grid service operation. Grid service instances should apply authorisation policy on all existing operations. OGSA defines standard interfaces for remote management of access control policy dealing with operation authorisation management and subject equivalency. Again within this approach a possible security threat lies with notification interfaces providing a notification source for client subscription. This generates one or more notification generators such as notification messages, filtering, topics third party filter services among many others since a wide variety of uses are possible. The OGSA philosophy thus requires partnerships for service development. We analyse the Alliance security framework from [4] that we propose as a new emerging technology to be incorporated in such relevant partnerships.

In this context we focus on four security properties a system should offer for applications dealing with confidential data.

1. *Data Privacy* not only during transport but also on the nodes.
2. *Code Integrity* to ensure that each node gets a correct code.
3. *Execution Integrity* to ensure that exactly the code that is supposed to be executed is actually executed.

4. *Robust Computations* that tolerate some inputs from malicious nodes.

The last three properties directly lead to the correctness of a wished functionality (like the simulation of an aircraft). Since data transfer is secured by symmetric encryption most damage can be done on the nodes themselves. They are not supposed to be trusted. Trust is only assumed via Grid operators who can be held to account.

By incorporating secure agent Alliances, we profit from the security guarantees of this model and gain a Grid in which nodes do not have to be trustworthy and computations remain confidential and correct as long as not more than a user-defined number t of nodes participating in the computations is malicious.

In this paper we will not go into technical details. It is a position paper in which emerging ideas from agent technology are presented which could lead to a secure Grid. It is structured as follows: In Section 2 we will review some properties of multi agent systems (MAS). We will mainly focus on security threats since they are transfered to the Grid when using agents within it. Section 3 presents the model for secure mobile multi-agent computation. Then, in Section 4 we will firstly discuss the use of the model in a static Grid model and secondly in a dynamic one. Finally, Section 5 discusses the results and future work.

2. Agents and Grids

Integrating agents into Grid technology is no new idea and has been mentioned in various publications with different motives. MARISM-A [13] uses mobile agents to look for anomalies, to monitor local activity or to perform track monitoring in widely distributed systems. D'Agents focuses on scalability of the number of agents per machine and the computing power of those machines, market-based resource control, disconnected operation and different information-retrieval techniques. Within this project a Grid Mobile-Agent System, which allows agents to migrate to different mobile-agent systems, has been proposed in [10].

The reasons for combining the techniques from Grid computation and agents lie in the capability of agents to allow network reduction, dynamic load balancing and asynchronous user-defined task-execution. On the other side, agents and especially *mobile* agents suffer from serious security threats which are transfered into the Grid, when integrating them. Besides attacks during transport a malicious node may attack an agent system by:

1. malicious routing,
2. violation of execution integrity,
3. violation of data integrity,
4. long-lasting and accumulating corruption of agents,
5. spying on sensitive data,
6. manipulation of the computed results and by
7. Denial-of-Service (DoS) attacks

As mentioned above, such threats cannot be tolerated in a Grid that is supposed to be used as testbed for new and valuable technologies. Therefore, it is of prime importance to integrate a sufficiently secured agent framework into a Grid. We will do so using agent Alliances from the model for secure multi-agent computation from [4] which is explained in the following section.

3. The Alliance Model

The work of Endsuleit and Mie in [4] is strongly motivated by secure multi-party computation (SMPC). The latter is thought to robustly implement a common function f (like playing a game) within a group of n fixed people (in the following they are called *parties*). Each party is possibly malicious, which means it is either a passive adversary by spying on data or doing some additional computations that are not part of the protocol or an active one who tries to damage/disturb the common computation. There have been a lot of hight quality publications on this field during the last two decades, such as [2, 9] for synchronous networks and [1] for asynchronous networks. All of them are based on Shamir's secret sharing scheme [14] which allows to share a secret like a private key or, like in our case, computational states.

The model from [4] seizes this method by constructing an Alliance of n agents to solve a common task instead of launching one single agent that is quite difficult to secure. All data is t-shared among the agents by using the verifiable secret sharing scheme from Ben-Or, Goldwasser and Wigderson [2]. This means that data reconstruction requires knowledge of at least $t + 1$ different shares. With less than $t + 1$ shares no information is revealed. Another property we gain from [2] the tolerance of common computations on shares against up to t malicious inputs. Both properties together lead to a system in which confidential and robust computation is possible.

The degree t of robustness is fixed by the size n of an Alliance because due to [2] the equation

$$t = \lceil n/3 \rceil - 1$$

must hold. This equation enables the user to determine a security level depending on real network assumptions for his application by wisely choosing the size of his Alliance. The construction of an Alliance of size $n = 4$ is illustrated in Figure 1. Due to the user interface the user still defines

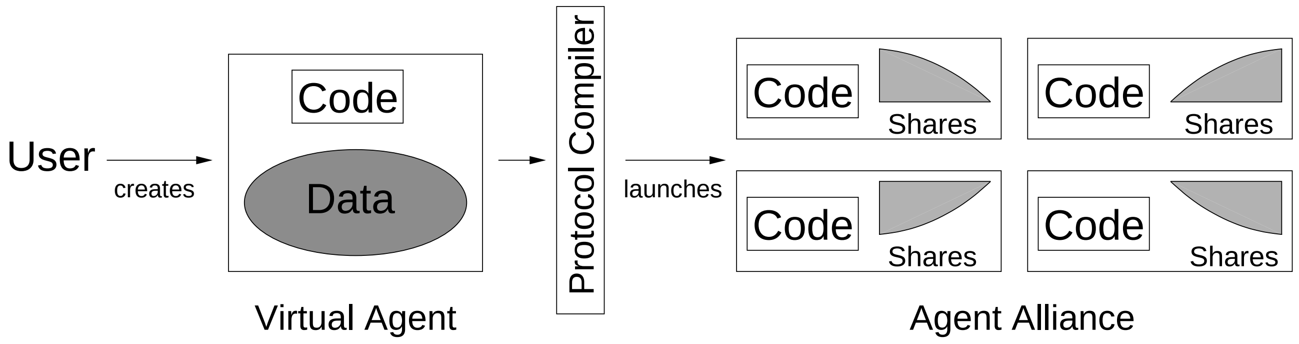

Figure 1. From Single Agents to Agent Alliances

one (virtual) agent that is capable to fulfil a special task. Internally, a protocol compiler transforms the agent's program into one that is capable to perform distributed computations on shares. Additionally, all data is split into four 1-shares. Finally, an Alliance of four agents is generated and launched.

Figure 2 continues the example of an Alliance of size 4 which has been sent to make a common computation on 4 fixed nodes. One of them, Node 4, is malicious and fakes his agent's shares. Since $t = \lceil n/3 \rceil - 1 = 1$ the computations of the Alliance are robust against one adversary. Consequently, the originator is able to re-construct a correct result upon result return.

The model focuses on four aspects:

1. *Securing Alliance computations:* In SMPC the n parties are fixed. Since we allow *mobile* agents we have to regard the hosting servers of an Alliance as the parties because they have full control over the agents. As long as no migration takes place, all computations are secured by SMPC. Those computations remain correct as long as not more than t hosts are malicious at any time.

2. *Use of a protocol for synchronous networks:* [2] assumes a synchronous network. Although an Alliance is supposed for asynchronous networks like the Internet, the protocol is used in the framework because it is more efficient than those for asynchronous networks. This is possible by enforcing a kind of synchronisation through timeouts.

3. *Integration of migration:* In SMPC all parties are fixed. Therefore, migration processes are not covered. The Alliance framework solves this problem by securing all involved computations and actions (new locations, actual hop, renewal of shares, etc) either through distributed computations or majority decision. In each migration new agents are created. The code of the new agents is fixed by a majority decision of the old agents. This is possible because all agents have the same code. The new agents' shares are generated with the method of [12]. In a *mobile* agent scenario this is absolutely necessary to prevent from an adversary collecting more than t valid shares.

   Thus, a corrupted agent does not survive a migration.

4. *Concurrent protocol execution:* To enable concurrent protocol execution on one host without loosing security guarantees the Alliance model is embedded into Canetti's model for universally composable security from [3]. The description of this model is out of scope of this paper and can be taken from [4].

3.1. Results

Thanks to robust computations and sharing its computational state, all but one security threats for agents mentioned in Section 2 do not exist for an Alliance. The remaining threat comes from real network and is the possibility of DoS attacks. This kind of attack is a current and unsolved problem for any computer system. Endsuleit and Wagner consider DoS attacks on agent Alliances in [5] and propose countermeasures which do not fully protect an Alliance but which make it possible either to detect the attack or to make its accomplishment much harder.

In principle, an Alliance may be defined in a way that no communication with a trusted party or the originator is necessary for means of security or functionality. It is then completely autonomous. An additional "feature" of the system is that computing on shares is like computing with encrypted data. Consequently, this is very interesting for enterprises which want to keep data input as well as intermediate data confidential.

A restriction definitely existing is that all security properties are depending on the upper limit of the number of malicious Alliance members. As soon as the adversary gets

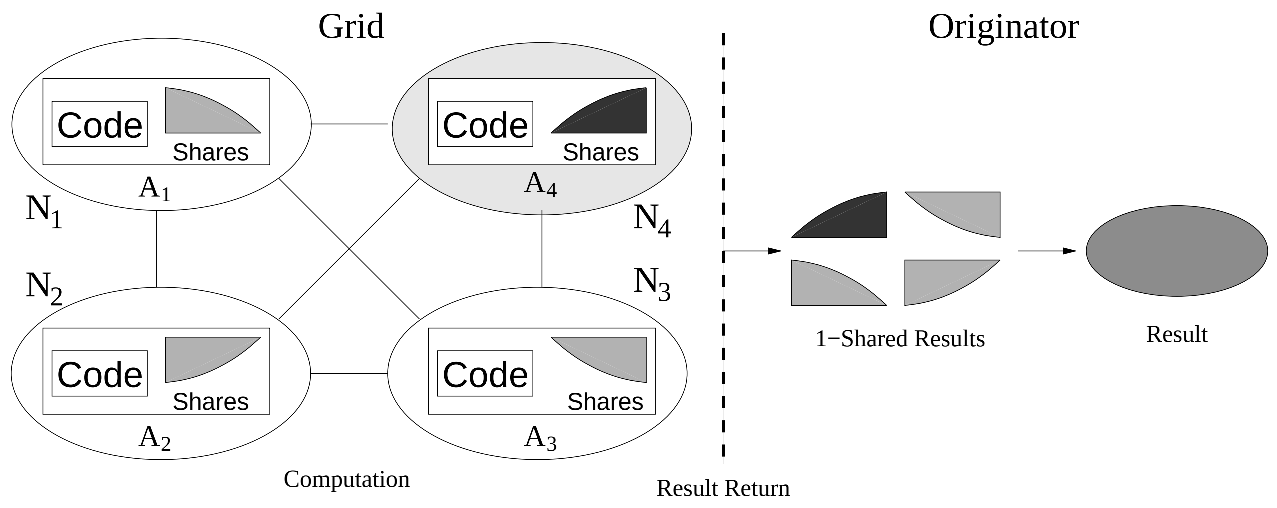

Figure 2. Result return with one malicious node

control over more than $t = \lceil n/3 \rceil - 1$ agents he gains full control over an Alliance. This risk can be taken as [5] has shown that it is due to the self-repairing possibilities of an Alliance not extremely high.

3.2. Efficiency

The model from [4] uses [2] for computations on shares which extensively uses broadcast channels and causes a communication complexity of $O(mn^6)$ (with n the size of the Alliance and m the number of multiplications needed). As a consequence the model is not practical. Hirt et al. presented in [11] a protocol for distributed computation with complexity $O(mn^2)$. Considering n agents these are linear costs for each hosting server. Therefore, [5] has inserted this protocol instead of [2] into the Alliance framework.

4. Alliances in a Grid

Now, lets turn to Grid. We assume data transfer to be encrypted and all Grid users and nodes to be authenticated. The latter is not necessary for the agent framework but nonetheless it is a feature that also simplifies the agents lifes (for a discussion on this topic see [5]). In addition, we require the nodes to host agents (whether member of an Alliance or not does not matter) within a sandbox without possibility to communicate with other agents hosted on the same node. Together with having the agents code signed by a trusted party it is valid to assume that there is little risk for a node executing an agent. Therefore, we consider this side as secured. We claim that by use of agents the Grid itself is not less secure than without agents.

As soon as code or data is transfered to per se untrusted nodes it is in danger, independently from being part of an agent or not. And, this is the point at which the Alliance framework using secure multi-agent computation comes into play.

Consider a function f, like a complex simulation, that shell be computed. All data needed from the originator of the Alliance is split into t-shares and distributed among n agents. The code which computes f is transformed in a way that for any computation n inputs are necessary and n outputs are produced (one for each Alliance member). It could also contain an interpreter that is able to read software modules from distributed computers and to act like the protocol compiler in Figure 1. As mentioned before, we consider the code to be the same for each agent to enable agent renewal via majority decision. But it could contain subtasks for each agent so that parallel fulfilment of the task is possible.

The resulting Alliance can be used in two different scenarios.

4.1. Static Applications

For applications in which the originator knows that data and computation resources necessary for the fulfilment of the given task are available on at most n Grid nodes, it is sufficient to consider the agents as static. In this case one could even think about refraining from agents, to consider each node as party participating in a common computation and to solely use the protocol for secure multi-party computation from [11]. In this case all computations are fully protected by the used protocol. Intermediate data and results are confidential and correct as long as not more than t nodes are malicious.

4.2. Dynamic Applications

The dynamic scenario is much more flexible since it allows agents to transfer from one node to another in order to fulfil their task. But the mechanisms from the Alliance framework *must* be included to secure the migration process. In this scenario it is possible to use Alliances for any thinkable task that requires a high security level.

In [5] has been shown that the probability of having more than t corrupted agents rises with each migration. Therefore, especially for applications that need a high number of migrations the Alliance size should be well chosen to offer sufficient redundancy.

5. Conclusion

In this section we will discuss the pros and cons (like any other system our agent framework has the latter, too) of using agent Alliances within a Grid.

The main advantages are:

1. To day, trust in open Grids is based on keeping Grid operators responsible for malicious nodes. In practice this may be sufficient for most applications but the more money is involved into the application the higher is the risk of having malicious nodes. Let's say, if I succeed in stealing 1 Billion Dollar by betraying once, I will not really worry about being excluded from the Grid afterwards.

   An agent Alliance can tolerate up to t malicious nodes without *any* loss of security, and consequently trust is not needed.

2. Secure multi-agent computation allows applications containing confidential data that require the highest security level. As long as no more than t agents are corrupted at any time *no* information is revealed.

3. Agent Alliances are completely autonomous and do not need any communication with a trusted party or the originator.

4. Sharing all data enables an Alliance to perform very sensitive operations like using a shared private key for contract signing.

5. Agent Alliances can be combined with other techniques. A node does not have to adapt to the framework and e.g. split its data into shares. In fact, we do not demand to perform *all* applications with Alliances. They are simply a very good tool for some special applications. For general use in a Grid they are not efficient enough.

6. In our discussions we always use an Alliance to fulfil a user-defined task. But is is also thinkable to define an Alliance for intrusion detection or managing tasks within a Grid. Indeed, using single agents for such tasks is very dangerous because in case of being corrupted they could be able to make a kind of DoS attack on the system by pretending the system being attacked and initiating countermeasures. In an Alliance the decision about the existence of a threat would be made jointly.

7. In case of building the whole Grid on Alliances, concurrent protocol execution is still secure because the framework is embedded into Canetti's work. Most people are not aware that in case of concurrent execution of secure protocols one protocol could leak information about another one and security is in danger.

Besides all those advantages we do not want to hide the few (but sometimes important) restrictions of the model.

Main object in discussions about SMPC is and has ever been the complexity of those protocols. As mentioned before, the one from [11] is ”only” quadratic in the number of Alliance members, but for practical deployment such a complexity has often been the fall for an algorithm. Therefore, we explicitly do not propose our framework for general use, but only for sensitive applications.

Besides this, agents are hard to protect, therefore we sternly advise to follow our proposal in Section 4.1 and to use SMPC without agents when the application allows this.

References

[1] Michael Ben-Or, Ran Canetti, and Oded Goldreich. Asynchronous secure computations. In *Proceedings of 25th Symposium on Theory of Computing (STOC)*, pages 52–61, 1993.

[2] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *Proceedings of 20th Symposium on Theory of Computing (STOC)*, pages 1–10, 1988.

[3] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. Report 2000/067, Cryptology ePrint Archive, 2000.

[4] Regine Endsuleit and Thilo Mie. Secure multi-agent computations. In *Proc. of Int. Conf. on Security and Management*, volume 1, pages 149–155. CSREA, 2003.

[5] Regine Endsuleit and Arno Wagner. Possible attacks on and countermeasures for secure multi-agent com-

putation. Internal Report 2004-7, University of Karlsruhe, 2004. Submitted to SAM'04.

[6] Ian Foster, Carl Kesselman, and Steven Tuecke. The anatomy of the grid — enabling scalable virtual organizations. *International Journal of High Performance Computing Application*, 15(3):200–222, 2001.

[7] Ian Foster and Carl Kesselmann, editors. *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann, 1999.

[8] The Globus Toolkit. Available in electronic form under `www.globus.org`.

[9] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game. In *Proceedings of 19th Symposium on Theory of Computing (STOC)*, pages 218–229, 1987.

[10] Arne Grimstrup, Robert Gray, David Kotz, Thomas Cowin, Greg Hill, Niranjan Suri, Daria Chacn, and Martin Hofmann. Write once, move anywhere: Toward dynamic interoperability of mobile agent systems. Technical Report TR2001-411, Dept. of Computer Science, Dartmouth College, 2001.

[11] Martin Hirt and Ueli Maurer. Robustness for free in unconditional multi-party computation. In *Proc. of Advances in Cryptology - CRYPTO 2001*, volume 2139 of *LNCS*, pages 101–118. Springer Verlag, 2001.

[12] Rafail Ostrovsky and Moti Yung. How to withstand mobile virus attacks. In *Proc. of the 10th Ann. ACM Symp. on Principles of Distributed Computing*, pages 51–59, 1991.

[13] Sergi Robles, Joan Mir, Joan Ametller, and Joan Borrell. Implementation of secure architectures for mobile agents in marism-a. In *Mobile Agents for Telecommunication Applications (MATA)*, volume 2521 of *Lecture Notes in Computer Science*, pages 182–191. Springer Verlag, 2002.

[14] A. Shamir. How to share a secret. *Communications of the ACM*, 22:612–613, 1979.