

# Issues in Building Intrusion Tolerant Group Membership Protocols

*Narasimha Prasad Subraveti, Soontaree Tanaraksiritavorn, and Shivakant Mishra*

*Department of Computer Science*

*University of Colorado, Campus Box 0430, Boulder, CO 80309-0430, USA.*

Email: [subravet | tanaraks | mishras]@cs.colorado.edu

## Abstract

Intrusion tolerant group membership protocols constitute an important part of an intrusion-tolerant group communication system. These protocols maintain a consistent system-wide view of correct group members in the presence of malicious failures. The paper provides a detailed analysis and comparison of current, intrusion-tolerant group membership protocols, identifies their strengths and weaknesses and identifies some of the critical issues that have not yet been addressed.

**Research Area:** Parallel and Distributed Systems Software

## 1. Introduction

With recent focus on building highly available distributed applications, intrusion tolerant systems are the need of the hour. A Group Communication Service facilitates this by providing Service Replication. A Group Membership Protocol (GMP) constitutes an important and complicated part of a group communication system. Informally, this protocol maintains a system-wide view of those group members that are functioning correctly and those that are not. It is complicated to build a correct group membership protocol due to the asynchronous nature of most existing distributed computing environments.

There is an urgent need to design and implement intrusion-tolerant group membership protocols that can work correctly despite Byzantine failures. Byzantine failures belong to the class of unrestricted failures in which failed processes may exhibit malicious behavior including a possible collusion to bring down the entire group communication system.

The main goal of this paper is to provide a clear understanding of the issues involved in building an intrusion-tolerant group membership protocol. Several research efforts to understand, design and implement an intrusion-tolerant group membership protocols are currently underway. These include [1, 4, and 5]. This paper reviews these research efforts, analyses them, and provides a comparison in terms of their advantages and disadvantages. Finally, the paper identifies some of the critical issues that have not yet been addressed.

## 2. Group Membership Protocol

A GMP is a protocol by which a set of processes reaches an agreement on the membership of the group. A process that does not comply with the operational conditions of the group/crashes/fails to transmit messages due to disconnection is deemed to have failed and hence is removed from the group. There is a possibility that the group member may recover from a crash or a communication failure and would like to rejoin the group. A group member may turn malicious intentionally (e.g. a disgruntled fired employee plans to inflict damage on the company) or unintentionally (A Trojan horse affected process) at any time during the operation of the group. These are some of the conditions that need to be handled by the GMP. It is essential for the membership protocol to ensure that any changes that occur to the membership protocol are correct and consistent.

### 2.1 System Model

A GMP makes the basic assumption that a group is formed by a set of processes after exchange of agreement messages. We assume that a typical group membership protocol consists of some countable, possibly infinite number of processes  $p_0, p_1, \dots$  that constitutes the group. At any particular point of execution, only a finite number of processes are present. A process is regarded correct if it behaves according to its specification. A faulty process can behave arbitrarily (Byzantine failures). Communication is mostly asynchronous. It is also assumed that the network does not partition. Each process has a failure detection mechanism that may suspect another process as either being faulty or correct. These suspicions can be false and may also differ for different processes. The set of identifiers of processes that are part of the group at a particular point of time is called a view/configuration. We install new views/configurations on change in membership of the group. These views are installed in increasing order and protocol execution is on a per-view basis. Each membership protocol assumes that the number of corrupt or faulty processes in a view must be less than  $1/3^{\text{rd}}$  of the total number of processes in the system. In the presence of Byzantine failures, this is a fundamental requirement for Group membership protocols to observe. Let us assume that  $k = \lfloor (n-1)/3 \rfloor$  and  $f = \lceil (2n+1)/3 \rceil$ .

## 2.2 Group Membership Protocol Properties

The goal of a group membership protocol is to enable a set of correct processes to agree on the group of processes that are members of the group. A GMP has to provide the following abstractions as a part of its protocol:

**Join Protocol:** A new process wants to join the group.

**Remove Protocol:** A faulty/corrupt process must be removed from the group.

Typical properties that a GMP [1, 4 and 5] satisfies are:

**Agreement:** If  $p$  and  $q$  are two correct processes, then both the processes will have the same membership view.

**Validity:** If a correct process  $p$  installs a view  $V_x$ , then  $V_x$  includes  $p$ .

**Integrity:** If a view includes a process  $p$ , but the subsequent view excludes  $p$ , then  $p$  was suspected by at least one correct member.

**Liveliness:** If there is a correct process  $p$  that is a member of a view and is not suspected by 'f' correct members of the view and there is a process  $q$  that is suspected by  $k+1$  correct members of that view, then  $q$  is eventually removed from the group view.

**Total Order of Views:** If correct processes  $p$  and  $q$  both install views  $V_x$  and  $V_{x+1}$ , then  $p$  installs  $V_{x+1}$  after  $V_x$  if and only if  $p$  installs  $V_{x+1}$  after  $V_x$ .

## 2.3 Fault Detection

A fault detector raises a suspicion event, suspecting failure of a specific group member. A GMP is typically invoked when the underlying fault detector raises a suspicion event. Two failure models have been considered in building group communication systems: Crash failures and Byzantine failures. In an asynchronous distributed computing environment, it is impossible to differentiate between the failure of the underlying communication mechanism and the failure of a process. Byzantine faults are tougher to handle than crash failures because it precedes the assumption that the process affected with a Byzantine failure can behave arbitrarily. The need to tolerate Byzantine failures introduces several new challenges in building a group membership protocol.

## 3. Analysis of Group Membership Protocols

We analyze and compare the three group membership protocols that are intrusion tolerant i.e. Rampart [5], SecureRing [1] and ITUA [4].

### 3.1 Rampart

Rampart is the first GMP that tolerates malicious intrusions using the technique of state machine replication to achieve high availability. The Rampart GMP uses a 3-phase commit protocol and provides strong consistency guarantees. In trying to achieve this, Rampart system excludes detection of corrupt members i.e. it relies on an

external fault detector. Public Key Cryptosystem is employed using Cryptolib package. Rampart also suggests the use of the concept of short term keys to optimize signature costs especially when group membership changes are infrequent. Rampart makes the following assumptions on its communication framework: 1) Reliable, Timed Asynchronous Communication model 2) Process-Process Communication takes place over Authenticated, Integral FIFO channels. Rampart adopts a Manager-based protocol structure where the manager is member with the highest rank at each correct process belonging to the view. Manager is responsible for suggesting an update to a view based on group member recommendations. If manager is faulty then protocol ensures that manager is removed from the group and a deputy protocol [5] is initiated to install the deputy as the new manager. The GMP begins when a process 'p' suspects another process 'q' to be faulty. It sends out a notification to the manager who waits till he receives 'k' notifications. Manager sends a suggestion for updating q's membership to all processes of the view. The suggestion embeds the notifications sent by the processes of the group. The manager waits for 'f' ACKs assuming that the correct process always ACKs one suggestion from the manager. Manager sends a proposal containing these ACKs to the members of the view. Each process in the view receives the proposal, verifies if it is created correctly and if so returns "Ready" to indicate its willingness to commit the update. Once the Manager receives 'f' Ready messages, then manager broadcasts the Commit message to all the processes in the view.

### 3.2 SecureRing

SecureRing employs the logical token ring architecture to solve the problem of group membership. The primary motivation of SecureRing group communication system [1] is to reduce cost of digital signatures. System is partially synchronous i.e. local non-synchronized clocks are used for processing and message transmission timeouts. Each and every processor D-multicasts messages to all other processors. Communication between processors is unreliable. No assumption of communication channels being FIFO or authenticated is made. The system builds a new Configuration if there is a persistent communication fault. The System relies on its message diffusion protocol to achieve message multicast. A processor cannot send messages until it possesses the ring token. The token is passed around the ring so that each processor gets its share of the channel in a round robin fashion. Public Key Cryptosystem is employed using Cryptolib package. Protocol uses packing of 'n' messages into a packet to allow a single digital signature to cover multiple messages. This implies avoiding multiple message exchange and achieves high throughput and reasonable latency. It uses message digests to achieve authentication and integrity. SecureRing integrates an

unreliable Byzantine fault detector to solve the problem of consensus in a Byzantine environment. The GMP begins when join messages are generated in the system asynchronously i.e. any correct member of the ring can generate a join message so as to initiate the GMP. Any processor can send a Join message if its fault detector raises a suspicion. Any correct processor that is part of the group can initialize the GMP once it receives 'k' Join messages by the message diffusion protocol. A processor achieves agreement when the difference `my_proc_set - my_fault_set` contains at least 'f' processors and the processor has recorded this in its agreement array. The processor enters the Commit state where a "Commit" token is circulated and every processor D-Receive it from each member of the new ring. The members then enter the Recover state where messages can be used to retransmit messages and tokens from the old ring as necessary without generating any new messages

### 3.3 ITUA

ITUA project [4] had the goal to develop a middleware based intrusion tolerance solution that helps building survivable distributed applications. The GMP system model is exactly similar to Rampart except that it does not assume a FIFO and reliable communication channel. It uses a manager based protocol structure like Rampart where the manager is termed leader (lowest ranked process). It introduces flexibility in its fault detection mechanism by incorporating an interface Suspect (process rank I, reason R) which can be invoked by any member in the group communication protocol stack. The GMP is initiated by any process that invokes the Suspect function. The GMP starts when a leader (who is responsible for initialization of GMP) on receipt of  $k+1$  suspicion messages multicasts "New View" messages to all the group processes. The process that generated the suspicion changes to the first phase. All other non leader processes change to wait phase. The leader waits for  $2k+1$  "Ack-New View" messages for the "New-View" messages it sent and issues a "Commit" message which implies that all processes in the group are ready to switch to the new view. When a process receives a valid "Commit" message, it broadcasts a signed "Ready to Switch" message to all the correct processes. Then the processes enter the message stabilization phase which ensures that all the correct processes broadcast the same set of messages in the current view. Each of the three phases of view installation has timers to ensure liveness.

## 4 Comparison of Group Membership Protocols

**Architecture:** The structure of a group membership protocol can determine the complexity of the message delivery mechanism of the underlying group communication system. Rampart and ITUA Systems assume a flat architecture. The Secure Ring protocol

introduces the concept of treating the process group as forming a logical ring structure. This ensures fairness by using a token ring mechanism of message multicast.

**Communication Model:** Rampart and ITUA build their group membership protocol on a timed asynchronous communication infrastructure where there are no time bounds. They use timeouts for detecting performance failures due to message transmission and scheduling delays. The SecureRing Protocol is partially synchronous which means that there are bounds on processing and message transmission times once the system is stable. Processors have access to local clocks but the clocks are not synchronized. All the three protocols provide reliable and total ordered multicast message multicast.

**Communication channels:** Rampart is based on the assumption that the network provides a FIFO and authenticated point-to-point communication channel between each and every pair of processes. ITUA assumes an unreliable channel and relies on its multicast protocol to provide FIFO delivery. Secure Ring employs a more practical approach that it does not make any of the above assumptions, i.e. it assumes that the channel is not reliable and there is no explicit ordering of messages in the FIFO stack.

**Security Model:** All the three protocols assume the existence of a Public Key Cryptosystem:

- Each process/processor has a unique private key that cannot be stolen to digitally sign messages.
- Each process/processor can obtain the public key of other processes/processors (as and when required) to verify signed messages.

SecureRing and ITUA protocols employ collision and inversion resistant message digest functions like MD5/SHA-1. Rampart uses an interesting feature to optimize signature creation. Assuming view changes are infrequent, each process uses a different short-term public key pair in the echo multicast protocol (reliable multicast when there are no membership changes) for each view. In its first echo multicast, each process piggybacks a new short-term public key on the message being multicast. Till the multicast is stable, the process uses the long-term private key for signing echoes for that view. It switches to a short-term private key corresponding to the public key that it multicast once the multicast is stable. To limit the duration of short-term keys, the membership protocol generates a "null view" change (where the next view is identical to the current view) message periodically.

**Group Formation:** Most group communication protocols use the GMP to determine membership with the assumption that a group of processes are already created. Rampart and SecureRing follow this model. ITUA protocol has a mechanism that illustrates how groups are

formed. Each process is given the following: A Process group identifier, number of tolerable malicious intrusions list of certified Public Keys. A single process forms a singleton group. This will unreliably broadcast a gossip message indicating its public key and the process group it desires to join. One of the constituent processes will be started with a singleton group identifier that is the same as the identifier of the intended process group. This process will be the nominated leader process. Once it receives a gossip message, it checks message validity and form a new view. The group is completely formed when the number of processes in the group has reached the desired intrusion tolerance.

**View Installation:** All the three protocols employ the 3-phase commit strategy. The first phase is the Consensus phase where the protocols try to reach consensus on the suspicions generated by the respective fault detectors. The second phase is the Agreement phase where the protocols reach agreement on the next view to be installed. The final phase is the Commit phase where the view is updated and the system is verified for consistency. The SecureRing Protocol uses the terms Gather, Commit and Recover for the three phases. The Gather phase is analogous to performing the first two phases mentioned above. The Recover phase is used for the processor to deliver a configuration change message to the application and ensure system consistency and stability. The ITUA protocol incorporates two wait phases in order to handle multiple simultaneous faults which may occur anytime during the view installation procedure.

**GMP Initialization:** A GMP is initiated when there is a change of phase from the initial operational state it is in. In Rampart, a process suspects another process in its view. It notifies its suspicion to the manager. The manager collects notifications from  $k+1$  group members. The manager then initiates the GMP for view update. Processes do not change state till the manager issues a suggestion for a view change. In SecureRing, the representative (manager) process does not play an active role in protocol initialization. A processor receives  $k+1$  Join messages by the message diffusion protocol. The processor shifts to the next state (Gather) and the GMP is initiated for creating a new group. The process changes to the state only after receipt of  $k+1$  Join messages. In the ITUA GMP, a process suspects another process in its view. It changes its state to the first phase, broadcasts suspicion to the group. A non-leader process that has received  $k+1$  messages goes to the first Wait Phase and times the response from leader. The Leader broadcasts "New-View" Message which contains the new view and also reason for the change in view i.e.  $k+1$  "Suspect" messages.

**Group Membership Agreement:** Once the group membership protocol is initiated then the processes have

entered the agreement/gather phase where it waits for confirmation of change in group view. Rampart performs this as follows: The Active manager needs to receive 'f' acknowledgements to proceed to send a proposal to all the members of the group indicating that the majority of the processes agree on the new view. In SecureRing, a processor achieves agreement when `my_proc_set-my_fault_set` contains atleast 'f' processors and when the processor has recorded as reaching agreement (in its agreement array) every processor in `my_proc_set-my_fault_set`. In ITUA, the leader waits for  $2k+1$  "Ack-New-View" messages for the "New-View" messages it sent before it issues a "Commit" message.

**Group Membership Commitment and Stabilization:** Once the group membership protocol has agreed on the new membership then we need to commit the new view. In Rampart, the manager on receipt of the "Ready to Commit" messages from 'f' processes commits the new view. The manager then broadcasts the "Commit" messages. In SecureRing, the processor enters the Commit state where a "commit" token is circulated and every processor receives it by diffusion from each member of the new ring. The members then enter the Recover state. In the Recover state, we can retransmit messages and tokens from the old ring as necessary. No new messages are generated in this stage. In ITUA system, the Leader issues a "Commit" message which implies that all processes in the group are ready to switch to the new view. When a process receives a valid "Commit" message, it then broadcasts a signed "Ready-to-Switch" message to all the correct processes. After this, the processes enter the message stabilization phase which ensures that all the correct processes broadcast the same set of messages in the current view.

**Fault model:** All the three protocols provide mechanisms to handle Byzantine fault tolerance and persistent communication faults with the presence of a fault detection mechanism as a part of the GMP or as a separate entity closely coupled with the GMP or as a universal interface. The fault detection model of all the three protocols assume that a process that is corrupt cannot be reinstalled into the group with the same process identifier and its signature. Once a process is detected as corrupt, then it is corrupt forever. SecureRing protocol handles persistent communication faults by using timeouts on message transmission and delivery. This means that a process/processor suspected faulty due to a transient communication failure is relieved of the suspicion once the communication between the processes involved is restored.

**Fault Detector Existence:** A Byzantine fault detection mechanism can be achieved by incorporating a separate fault detector that couples with the GMP and multicast protocol. This technique is employed in Rampart.

SecureRing integrates an unreliable fault detection mechanism with its GMP and multicast protocol. The advantage of having such a system is that the functioning of GMP is largely influenced by the fault detector. So there is an inherent tight coupling between the GMP and the fault detection mechanism. This comes at the expense of flexibility. The advantage of having an unreliable fault detector as a separate entity makes the Group Communication System modular. ITUA takes the middle approach (adding flexibility) in that it incorporates an interface called Suspect (process rank I, reason R) which can be invoked by any protocol in the group communication stack. This means that the invoker can be an external third party fault detector also.

**Multiple Fault Handling:** Rampart handles multiple faults in a FIFO manner which is not ideal especially in the case of malicious processes. SecureRing provides an architecture that allows handling multiple faults that are injected simultaneously into a system. Multiple simultaneous faults are handled by broadcasting new Join messages while there is a change in either `my_fault_set` or `my_proc_set`. This is perceived as a change in the configuration and all the processors shift to the Gather state irrespective of their current state. ITUA architecture also provides an efficient mechanism for handling multiple faults that occur at the same time by broadcasting “Need more Change” messages asynchronously and changing state to Wait Phase 1 and resuming the GMP from that state.

**Message Complexity:** Let us consider message complexity based on two cases:

**Best Case:** *Assumption:* We assume a single fault (non manager) in an ‘n’ process system.

The cost of Rampart system for the best case during the initial agreement stage is ‘k’ messages. The manager then sends n-1 Suggest messages. On receipt of ‘f’ acknowledgements, the manager sends n-1 “Ready to Commit” messages, then manager waits for ‘f’ “Ready to Commit” acknowledgements and in the last stage sends “Commit” message to n-1 processes. This gives an overall complexity cost of nearly 6n i.e.  $O(n)$ . The cost of SecureRing is based on every processor relying on the diffusion multicast protocol. It takes k\*k messages for each processor to shift to Gather state. During the Gather stage each and every process multicasts f\*f messages to reach agreement on commitment. Finally the “Commit” messages are sent to (n-1)\*(n-1) processes. Hence ITUA incurs a cost that is similar to Rampart except for the state where it waits for “Commit” from all correct processes. From the leader’s point of view, it requires ‘k’ messages to reach agreement on initialization of membership change. The leader then multicasts the “New-View” messages to n-1 processes. The protocol then waits for 2k+1 “Ack-New-View” messages before moving to the Commit phase. The protocol then sends “Commit”

messages to n-1 processes. Hence message complexity is 5n i.e.  $O(n)$ .

**Worst Case:** *Assumption:* We assume ‘k’ processes/processors are faulty.

The manager can also be faulty or corrupt. This means we have ‘f’ correct processes. The cost of Rampart system for the worst case during message exchange is k times the best case complexity. This gives an overall complexity cost of nearly  $6n^2$  messages. The cost of SecureRing is based on every processor relying on the diffusion multicast protocol. The worst case complexity occurs when a new configuration is about to be committed and a fault is suspected resulting in a new join message. This forces all processors of the system to go to gather state. If this happens for all the ‘k’ processes; then the worst case cost is ‘k’ times the best case cost. Hence it is of order of  $n^3$  i.e.  $O(n^3)$ . ITUA is more efficient than Rampart as its worst case is not as bad as Rampart. The worst case occurs when a new faulty/corrupt process is found just before the new view is committed. From the leader’s point of view, it requires ‘k’ messages to reach agreement on initialization of membership change. Then the leader multicasts the “New-View” messages to ‘f’ processes. The protocol then waits for 2k+1 “Ack-New View” messages before moving to the Commit phase. The protocol then sends “Commit” messages to ‘f’ processes.

## 5 Discussion

Intrusion-tolerant group membership protocols must deal with two difficult issues. First, they need to deal with the difficulty of detecting Byzantine failures. In particular, group membership protocols are time consuming and any service implemented on top of group communication system becomes unavailable while the underlying group membership protocol is in progress. As a result, these protocols must ensure that they are invoked only when it is fairly certain that the security of a member has been compromised. Second, they must ensure that a compromised group member gets as little time as possible to launch malicious attacks on the group communication systems i.e. a compromised group member should be removed from the group as soon as possible, so that it does not get much time to inflict damages in the group communication system. These two issues are contradictory in nature. The first issue entails that the group membership protocol be invoked only after it is fairly certain that the security of a group member has been compromised. However, because of the inherent difficulty in detecting Byzantine failures, it may take a relatively long period time to be fairly certain that the security of a group member has been compromised. Thus, the first issue essentially results in delaying an invocation of a group membership protocol. The second issue, on the other hand, entails that a group membership protocol be invoked as soon as there is even a slight suspicion that a group member is faulty. All three current intrusion-

tolerant group membership protocols [1, 4, and 5] address the first issue. The group membership protocol in all these systems is invoked only after a group member has been suspected to be faulty by at least  $1/3^{\text{rd}}$  of the group members. As a result, all these approaches suffer from a vulnerability in which a compromised group member can launch malicious attacks for a relatively long period of time. An important criterion that must be considered in designing intrusion-tolerant group membership protocols is how to provide a reasonable balance between these two issues.

Failure detectors play an important role in determining the effect of a group membership protocol. In the current approaches, a failure detector is either completely independent of the rest of the group communication system [5], or tightly integrated with atomic broadcast and group membership protocols [1]. The main advantage of the former approach is that a failure detector can be used with any atomic broadcast or group membership protocols. The obvious disadvantage is that this failure detector cannot exploit any semantic information related to the group communication system in detecting failures. On the other hand, the main advantage of the latter approach is that a failure detector can make use of all the semantic information related to the group communication system in detecting failures. However, this failure detector is highly dependent on a specific atomic broadcast or group membership protocol. As a result, it cannot be used in conjunction with other atomic broadcast or group membership protocols, and any changes in the atomic broadcast or group membership protocol may necessitate changes the failure detector. An important criterion that must be considered in designing intrusion-tolerant group membership protocols is how to build a failure detector that can exploit the underlying group communication semantics, and at the same time be sufficiently independent of the atomic broadcast or group membership protocols being supported. At present, we are investigating the design and implementation of an intrusion-tolerant group membership protocol addressing both of these criteria. An example prototype of such a mechanism has been provided by considering the fault detection mechanism as an interface [4].

All the three protocols are initiated by a protocol initiator (either the manager/any other group member), depending on the protocol chosen. Initialization is done only after the protocol detects that at least  $1/3^{\text{rd}}$  of group members suspects a particular group member. Then, all three protocols enter phase 1, consensus phase. In this phase, group members try to reach agreement on the new view by exchanging their perspectives about what should be the new view. Rampart and SecureRing reach this agreement when the condition is  $\lceil (2n+1)/3 \rceil$ , where  $n$  equals to the number of current group members. ITUA uses

different criteria i.e.  $2k+1$ , though  $\{2(n-1)/3\} + 1$  mathematically equals to  $\{(2n+1)/3\}$ , the ceiling and floor mathematical functions make  $\{2(n-1)/3\} + 1$  equal to  $2k+1$  only when  $n = 3j+1$ ,  $j = 1, 2, 3, \dots$ . The condition used in ITUA leads to fewer messages to reach agreement when  $n \neq 3j+1$ ,  $j = 1, 2, 3, \dots$ . Since we assume that there can be at most one-third of members ( $k$ ) failed, we can decide if the majority of group members agree upon a statement when one more than twice the number of supposedly failed group members expresses the same information. Thus, the condition posed by ITUA is sufficient to satisfy the requirement of this step.

To tolerate malicious failures, we need a mechanism to authenticate members and provide security over the communication channels. Cryptographic cost in membership protocols are expensive irrespective of the environment it runs on. In order to reduce cost, several systems implement it with some additional assumptions. Rampart suggests of using lower-bit modulo for encryption using short term keys. SecureRing uses the "message packing" technique. However, this technique cannot be used with some applications that required real-time encryption and response. We see that this technique is highly efficient if we thoroughly understand the application behaviors so we can properly adjust the parameters such as packages size and duration that applications can tolerate for delaying messages. ITUA does not suggest any technique to reduce its cost. We agree upon the necessity of the cryptography and optimistically believe that if we have incorporated this necessity into our membership design since the beginning, we would be able to reduce its cost.

## References

- [1] K.P. Kihlstorm, L.E. Moser and P.M. Melliar-Smith. The SecureRing Group Communication System, *ACM TISS 4(4)*, page 371-406, Nov 2001.
- [2] P.Pandey, Reliable Delivery and Ordering Mechanisms for an Intrusion Tolerant Group Communication System, MS Thesis, University of Illinois at Urbana-Champaign.
- [3] H.V. Ramasamy, et al. Quantifying the cost of providing intrusion tolerance in group communication systems. In *Proceedings of IEEE International Conference on DSN*, June 2002.
- [4] H.V. Ramasamy. Group Membership Protocol for an Intrusion Tolerant Group Communication System, MS thesis, University of Illinois at Urbana-Champaign, 2002.
- [5] M.K.Reiter. A Secure Group Membership Protocol. In *Proceedings of the IEEE Symposium on Research in Security and Privacy*, pages 176-189, 1994.