

Design and Implementation of a Secure Group Communication Protocol on a Fault Tolerant Ring

Özgür Sağlam^{1,2}, Mehmet E. Dalkılıç², and Kayhan Erciyeş²

¹ BEKO Elektronik R&D Center, Alsancak, İzmir, Turkey
OzgurS@beko.com.tr

² International Computer Institute Ege University Bornova, İzmir, Turkey
{dalkilic,erciyes}@ube.ege.edu.tr

Abstract. In this paper, we describe a secure group communication protocol for a fault-tolerant synchronous ring. Our protocol, named Secure Synchronous Ring Protocol (SSRP), integrates a secure group communication facility into an existing scalable, fault-tolerant ring protocol. SSRP is a hierarchical group communication protocol that employs Cliques GDH contributory key management protocol and Diffie-Hellman encryption algorithm. Security related functions and group communication functions are integrated into a single layer guaranteeing that not only the application messages of the group, but also the messages related to the group communication layer are protected. A novel fault-tolerant leader election algorithm and a proof of Virtual Synchrony semantics on SSRP are also given.

1 Introduction

Industrial and military applications that require the communication of information within a group are becoming an indispensable part of the modern computing. Security, in addition to reliability and high-availability, is one of the main characteristics of these group communication based applications. However, providing security for group communication is a complex task especially in the presence of network and node failures. The main issue in secure group communication is the key management. Since group has a variable view, changing dynamically as new members join or leave the group during the group's lifetime, a key management policy should have the ability to handle these dynamic operations. Moreover it should preserve the group communication secrecy independently from these changes. One approach relies on a single entity that is responsible for generating the group key and distributing it to the entire group. Such a scheme is open to the single point of failure problem. Kerberos [13] is an example of this scheme. Another key management approach selects a group member to generate new keys and distribute them to the other group members. This makes the system fault tolerant in the case of crashes and partitions since a new member will be selected to regenerate the group. Different from the above, there exists a contributory key management scheme. Steer et al. [11], Burmester and Desmedt [5], Group Diffie-Hellman (GDH) [12] and Tree-based GDH [8] key management protocols are examples of this scheme. All of these are based on 2-party Diffie-Hellman

key exchange protocol extended to the multi-party case. Contributory key agreement is suitable for peer to peer communication groups. There are some systems implementing security concepts over group communication environment. These are SecureRing [7] project at UCSB, the Horus/Ensemble [10] work at Cornell and SecureSpread [2] project at Johns Hopkins University. The Ensemble security work is the state of-the-art in secure reliable group communication and addresses problems such as group keys and re-keying [3]. Ensemble uses PGP authentication, which is not contributory. SecureSpread is a recent project on secure group communication where a secure communication layer between the Spread group communication daemon and the application layer is implemented. Different from Ensemble, it uses a group key structure that is contributory. The used structure is from Cliques Toolkit [12][8], using 2-Party Diffie-Hellman key exchange protocol extended to multi-party case.

In this study, we describe a system called Secure Synchronous Ring Protocol (SSRP) that relies on integrating a proven contributory key management protocol entity into the group communication protocol, which is developed by Tunali and Erciyeş [14][6]. This entity is in the communication layer and is completely independent from the layers above. The main goal of the work is to develop a fault tolerant, scalable, synchronous, real-time and secure distributed system. These properties allow the system to be suitable for large groups needing secrecy in the group messages. By integrating a security capability into the group communication layer, it is guaranteed that not only the application messages of the group, but also the corresponding information related with the group communication layer is protected from the outside attacks. The reduction in the total messaging needs to construct a secure group communication is another advantage in terms of bandwidth usage. In this study, only the attacks from the outside of the group are considered.

The rest of the paper is as follows. Section 2 outlines the hierarchical, fault-tolerant ring protocol [14] and the Cliques [4] protocol suite which is designed for group key management into dynamic groups. Section 3 presents the Secure Synchronous Ring Protocol (SSRP) with membership operations, leader selection algorithm, and the proof of virtual synchrony semantics. The conclusions are given in Section 4.

2 Background

The Hierarchical Fault Tolerant Ring Protocol is designed for distributed real-time systems [14] and it is a synchronous, hierarchical and scalable communication protocol that may be used for group management as shown in Fig.1. There are three entities within the protocol having dedicated responsibilities throughout this hierarchy named as *node*, *representative* and *leader*. Each sub ring (i.e. a cluster) consists of one controller called the *representative* and a limited number of *nodes*. Controller is a generic name for representatives at any level. Representative is responsible for connecting the corresponding cluster, called the *inner ring*, to the one step higher level ring called the *outer ring*. Also group membership operations of the cluster are its responsibilities. Representative releases an empty token at each period of time. The token passes through the ring and related fields are filled with requested information by the nodes and the token returns to the representative. The resultant token is passed to the higher

level cluster when an empty token of this level reaches the representative. Group communication processes in the higher level are the same with that of lower level. The controller of this level is called the *leader*. It has the same responsibilities on its cluster as the representative. This hierarchical structure based on nodes, representatives and leader can be built on n levels based on the group size and communication and computation overheads without loss of generality.

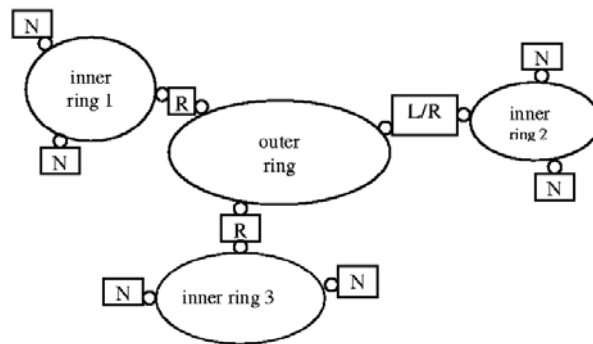


Fig. 1. The Ring Protocol with two level communication hierarchy [14]

It is possible to build various distributed applications on this architecture. At the lowest level, the distributed clock synchronization is built since the system model is synchronous. In the clock synchronization algorithm, the controller periodically requests clock information from the nodes and upon receipt of the clock information, the controller decides on the new clock value and dictates this to the whole system via the cluster representatives. Above the clock level, the distributed scheduler operates similar to the clock synchronization module [1]. In the group membership layer, the processes may belong to a particular group that is defined independent of the cluster topology. At each time period, a token is released to the first member of that ring by the controller. Node fills the token with the available information and passes it to the next member of the current view of the cluster. As long as a crash does not occur, the token returns back to the controller. In the case of failures, the protocol rebuilds the system and starts functioning again. When the token circulation is not finished in a certain time, then the controller starts to rebuild the view of the cluster by extracting the crashed node(s) by polling the current view. When the controller crashes, the partitioned cluster starts the leader election algorithm and determines the new controller of the cluster. Then, partitioned cluster rejoins the group through the new controller.

The membership management of this protocol is based on Virtual Synchrony model (VS) [15]. VS states that when a message is sent to the group, only the group members that are in the view of the group when the message was originally sent can receive the message. This property is a must for a group when building a strictly secure group communication system.

Cliques is a protocol suite designed for implementing group key management into dynamic groups. It consists of proven key management protocols each having different performance values [4]. Toolkit assumes an underlying reliable group communication system that provides reliable broadcast, message ordering and group member-

ship services. GDH [12] is one of the contributory key management protocols implemented in Cliques. It is computationally expensive but yields better performance in terms of bandwidth usage. It is also appropriate for ring based systems. GDH provides key independence and perfect forward secrecy and was proven secure to passive outside attacks. Active outsider attacks are not addressed in this protocol, however, active attacks can be prevented by the use of public key signatures. If all the group members know the public keys of each other, they can check the received messages by decrypting the signature with the public key of the expected sender. If signature is not verified, the message is simply ignored [4].

3 Secure Synchronous Ring Protocol (SSRP)

For SSRP, GDH protocol which has least bandwidth usage when handling group membership operations [4] is selected from the methods in the Cliques Toolkit. The management structure of the GDH is much like a ring application and GDH also supports handling the group key management operations via a controller. In the ring protocol, the group membership operations are performed independently in each ring. Similar to Iolus [9], the individual group keys are calculated by each ring. This provides scalability for large groups having hundreds of members. SSRP protocol calculates group key independently for each cluster while handling the group membership operations such as member join/leave, mass join/leave and group join/leave. The leader and the representatives are not only responsible for managing the cluster's membership operations but also managing the contribution process of GDH when generating the cluster key of each view. Here the inner ring messages are decrypted with the inner ring key and again encrypted with the outer ring key. However, this operation requires decryption/encryption of whole data which increases the computation costs. Instead, the approach proposed in [9] can be implemented in SSRP for this situation, since each cluster has independent group membership management [14].

3.1 Membership Operations

The following operations are provided by SSRP:

Member addition: The initial case is that there is only one entity waiting to add new members to the group. This entity is the leader. Leader accepts a finite number of members as representatives. After this, new joining requests by the nodes are served by the representatives until they reach a specific count. When joining the n^{th} member, total messaging need of this membership operation is $n + 3$.

Member leave: The GDH member leave protocol [12] is implemented without any modification. When a member leaves the cluster having n members, total messaging need of this membership operation is n .

Mass join: As suggested in [12], the chained implementation of controller based GDH single member join operation is used for mass join. When m new members join n member cluster, total messaging need of this membership operation is $n + 3m + 1$.

Mass leave: The GDH mass leave protocol [12] is implemented without any modification. When m members leave the cluster having n members, the total messaging need of this membership operation is n .

Group join and group leave: In SSRP, group join operations are carried out by the representative of the cluster. The group join operation is simply the member join operation of the representative into the outer ring controlled by the leader. Since each cluster has independent membership management, outer ring membership does not effect the inner ring operations. The same logic is applied to the group leave operation. In this case, member leave operation is applied on the representative of the cluster requesting to leave. Note that, since SSRP uses GDH for member addition, member leave, mass join and mass leave operations, the computation complexity and the security properties declared in [12] are preserved in SSRP.

3.2 The Representative Finite State Machine

The simplified finite state machine diagram of the representative for single layer is shown in Fig.2 with the following states:

Wait Period: In this state, the representative runs its timer in order to keep track of the token to be released for each data circulation period of the system.

Wait Data: The representative waits for the token to be filled by its group members in this state. If the filled token is received back in a bounded time, then the representative sets its state to *Wait Period* in order to start the next token circulation for the next period.

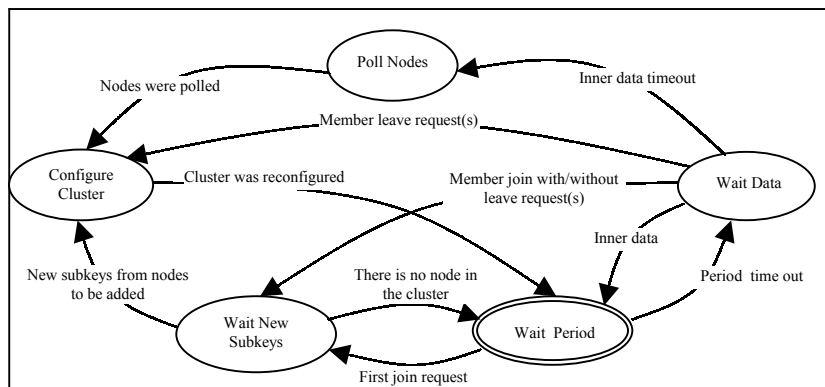


Fig. 2. Simplified FSM diagram of an SSRP representative

This is the normal communication loop of the SSRP. If the filled token is not received back in a bounded time, then the representative starts searching the ring to exclude the problematic node(s) from the group and sets its state to *Poll Nodes*. During the normal group communication operation, the new joining requests from the outside and the leave requests from the ring are held in request queues generated by the representative. If there are any join requests without leave requests in the request queues, the corresponding join protocol (mass or member join) is executed and the representa-

tive sets its state to *Wait New Subkeys*. If there are also any leave requests in addition to join requests then, first the subkeys of the leaving nodes are removed from the current subkeys list and the representative updates the resulting list with its new private key. After that, join protocol is started using these new subkeys to join the new members. In this case again, the next state is *Wait New Subkeys*. If there are only leave requests while in this state, then the representative starts the member/mass leave protocol. It removes the subkeys of leaving nodes from the current subkeys list and updates the resulting list with its new private key. The next state is set to *Configure Cluster*.

Wait New Subkeys: According to the controller based GDH member join and mass join protocols, controller of the group sends the last received subkeys that are defined in [12] to the new joining member(s) and it waits the updated values by the new members in the way that is defined in GDH protocol definitions. As the controller of this group, representative waits updated subkeys from the nodes to be added in this state. If these keys are not received within a bounded time, representative returns to the *Wait Period* state.

Polling Nodes: To find the crashed node(s), the representative polls the ring nodes and excludes the member(s) from the view which does not respond to this polling message within a bounded time. It removes the subkeys of the crashed nodes from the current subkeys list and updates the resulting list with its new private key. The next state is set to the *Configure Cluster*.

Configure Cluster: In this state, representative contributes its private key to the received or generated new subkeys. The resulting subkeys list is then replaced in a configuration token and sent to the ring members. Each ring member receiving this token then calculates the group key of the new group view that is to be shared among the ring members. Then the new group view is set up and the representative sets its state to *Wait Period* to run the SSRP protocol for this view.

3.3 SSRP Leader Election algorithm

Assuming the existence of a distributed clock synchronization service, we developed a leader election algorithm based on timers. Each node runs a timer checking the liveness of the controller. Each time a node receives a token or a polling message, it resets its timer since the origin of these messages in the SSRP protocol is always the controller. The first node that has a timeout will declare itself as the new controller. Timeout value, T_i which reflects node_{*i*}'s distance from the controller is different for each node. The Equation for T_i is given below where T_{clock} is the current clock value and $OFFSET$ is the expected time for the new controller to reconfigure the ring.

$$T_i = T_{clock} + (id_i * OFFSET) \quad \text{and} \quad OFFSET > \frac{nT_p + T_{config}}{2n}$$

3.4 Proof of Virtual Synchrony Semantics on SSRP

In this section, we show that the SSRP protocol preserves the virtual synchrony semantics that are pointed in [3] and defined in [15].

1. *Self Inclusion*: If a node installs a view v then it is a member of this view. Because the only way the node can receive the token t updating the group view is that its address should be known by the preceding ring member in order to send the token. This is possible if the representative places the information of this node into the current view since it controls the group view operations in SSRP.

2. *Local Monotonicity*: If a node installs a view v^+ after installing a view v then the identifier of v^+ is greater than the identifier of v . In SSRP, it is impossible to install a view more than once since the view installing operation is started by the representative once per view change occurring in the group.

3. *Sending View Delivery*: The new view installing operation is started only when the data token is received back by the representative at *Wait Data* state or at *Wait Period* state when there is no node in the cluster. So, if data is circulating through the ring, it is guaranteed that there will be no view change until the data is received by all the nodes that are members of the view in which the data has been sent.

4. *Delivery Integrity*: If a node receives a message from a token in a view, there is always at least one member, whose index number in the ring is lower than this node, that transmits this token in the same view before the node receives it. The reason is that the data is transmitted by tokens and tokens are always unicast among the ring members in one direction.

5. *No Duplication*: A message is not sent twice. Each token circulation has a unique number and the only source that creates this token is the representative.

6. *Self Delivery*: All ring members except the controllers (i.e. representatives and leader) has to direct the received token to the next member within a bounded time.

7. *Transitional Set*: In SSRP if a node n_1 and n_2 install a new view, then n_1 is in the transitional set of the n_2 if and only if n_1 's previous view is identical to n_2 's previous view. Consider the case that n_1 is not in the previous view of n_2 . This case is valid if n_1 is one of the new members requesting the join to the group since it is not possible to add n_1 into the current view without changing the view id. That is, the representative stops the token circulation and configures the view by adding the new members including n_1 so that the previous view of n_2 does not include n_1 . Then n_1 is not included in the transitional set of n_2 .

8. *Virtual Synchrony*: From sending view delivery property, in SSRP, tokens are released by the representative and they are received by all nodes of the view that tokens are sent in. Then, if two nodes are included in any two consecutive views they see the same set of messages in both views. All messages are delivered in total order since the messages are delivered by tokens circulating in one direction. So every member receives the messages in the same order.

4 Conclusions

In this work, we described a group communication protocol, SSRP, having a built in security implementation which maintains a group key shared among the group members for any membership operation. This protocol supports virtual synchrony semantics and has the security properties of GDH. Integrating GDH and the ring protocol does not increase the communication complexity of the resulting design compared

with the ring protocol and integration process is easy. This integrated protocol increases the system security by combining the security protocol and application data. Also we introduce a leader election algorithm for SSRP assuming clock synchronization service of the ring protocol. The protocol is fully fault tolerant in case of multiple crashes. Because adding security service does not degrade the communication performance, this protocol is suitable for wide area networks. SSRP protocol has the advantage of less bandwidth usage because GDH has low message complexity and also in SSRP, group membership and key generation operations are integrated, reducing the overall communication complexity. Hierarchical construction also makes this design suitable for large groups (> 100) since group operations are carried out in parallel within subrings. As a future work, we plan on improving the SSRP protocol by adding authentication service into the key management process.

References

1. Akay, O., Erciyès, K.: A Dynamic Load Balancing Model For A Distributed System. *Mathematical and Computational Applications*, Vol. 8 No. 3 (2003) 353 - 363
2. Amir, Y., et. al.: Secure Group Communication in Asynchronous Networks with Failures: Integration and Experiments. *Proceedings of the 20th IEEE International Conference on Distributed Computing Systems* (2000) 330–343
3. Amir, Y., et al.: Exploring Robustness in Group Key Agreement. *Proc. of the 21st Int'l. Conf. on Distributed Computing Systems --ICDCS'01* (2001) 399-408
4. Amir, Y., et al.: On the Performance of Group Key Agreement Protocols. *Proc. of 22nd Int'l. Conf. on Distributed Computing Systems --ICDCS'02* (2002) 463-464
5. Burmester, M., Y. Desmedt, Y.: A Secure and Efficient Conference Key Distribution System. *Advances in Cryptology – EUROCRYPT'94* (1994) 275-286
6. Erciyès, K.: Implementation of A Scalable Ring Protocol for Fault Tolerance in Distributed Real-Time Systems. *Proc. of 6th Symp. On Comp. Networks –BAS 2001* (2001) 188-197
7. Kihlstrom, K.P., Moser, L. E., Melliar-Smith, P.M.: The Secure Ring Protocols for Securing Group Communication. *Proc. of the IEEE 31st Hawaii International Conference on System Sciences*, Vol. 3 (1998) 317–326
8. Kim, Y., Perring, A., Tsudik G.: Simple and Fault-tolerant Key Agreement for Dynamic Collaborative Groups. *7th ACM Conf. on Comp. & Communication Security* (2000) 235–244
9. Mitra, S.: Iolus: A Framework for Scalable Secure Multicasting. *Proceedings of the ACM SIGCOMM'97* (1997) 277-288
10. Rodeh, O., Birman, K., Hayden, M., Xiao, Z., Dolev, D.: Ensemble Security. *Tech. Rep. TR98-1703*, Cornell University, Dept. of Computer Science (1998).
11. Steer, D., Strawczynski, L., Diffie, W., Wiener, M.: A Secure Audio Teleconference System. *Advances in Cryptology – Lecture Notes in Computer Science* (1990) 520-528
12. Steiner, M., Tsudik, G., Waidner, M.: Key Agreement in Dynamic Peer Groups. *IEEE Trans. on Parallel and Distributed Systems*. Vol. 11. No. 8 (2000) 769-781
13. Steiner, J.G., Neuman, C., Schiller, J.I.: Kerberos: An Authentication Service for Open Network Systems. *Usenix Winter Conference* (1988) 191–202
14. Tunalı, T., Erciyès, K., Soysert, Z.: A Hierarchical Fault-Tolerant Ring Protocol For A Distributed Real-Time System. *Special issue of Parallel and Distributed Computing Practices on Parallel and Distributed Real-Time Systems*, Vol. 2, No. 1 (2000) 33-44
15. Vitenberg, R., Keidar, I., Chockler, G.V., Dolev, D.: Group Communication Specifications: A Comprehensive Study. *Tech. Rep. CS0964*, Comp. Sci. Dept., Technion (1999)