システムソフトウエアと
オペレーティング・システム 60−16
マルチメディア通信と分散処理 61−16
（１９９３．７．９）

安全な放送通信プロトコル

三田 浩也、中村 章人、滝沢 誠
東京電機大学理工学部

グループウエアのような分散型応用では、複数エンティティ間でのグループ通信が要求される。本論文では、エンティティのグループを群とする。ローカルエリア網(LAN)や無線網では、媒体アクセス制御(MAC)層において放送通信が提供されている。放送通信網では、安全な群通信を提供する必要がある。本論文では、安全な群通信を提供するための方式を論じる。本プロトコルでは、非安全で高信頼な放送通信と公開鍵を利用して、群内の全エンティティだけが同一の秘密鍵を獲得できる。さらに、群内の一部のエンティティは、群内の他のエンティティが参加できないような(例えば、内証話など)安全通信を提供するための部分群の確立方式を論じる。

Secure Broadcast Communication Protocol

Hiroya Mita, Akihito Nakamura, and Makoto Takizawa
Tokyo Denki University
Ishizaka, Hatoyama, Saitama 350−0, Japan
e−mail {mita, naka, taki}@takilab.k.dendai.ac.jp

In distributed applications like groupware, group communication among multiple entities is required. The local area networks (LANs) and radio networks provide broadcast communication at the media access control (MAC) layer. The group communication among multiple entities can be easily realized by these networks. One problem in the broadcast network is how to provide secure communication for the group. In this paper, we discuss how to provide the secure group communication based on the distributed scheme by using less-secure broadcast networks. A group of entities is named a *cluster*. In the protocol, only and all the entities in the cluster agree on a same secret key by exchanging the nonce enciphered by the public key. Sometime, a subset of the cluster may require such secure communication that the other entities in the cluster cannot join the communication. In this paper, we discuss how to establish the secure subcluster communication in the cluster.

1 Introduction

In distributed applications like tele-conferencing and cooperative work [6], group communication [1, 11, 12, 13, 16, 17] among multiple communication entities is required. These applications may require secure group communication. In this paper, we would like to discuss how to support the secure group communication.

Since various kinds of communication networks have been widely used on the basis of the open systems architecture [7, 8], various kinds of computer systems can be easily connected by the communication networks. One problem in these systems is how to protect the system from attacks of malicious entities. Malicious entities can easily send and receive data units by using the broadcast networks. One solution is to encipher the data units by using a secret key and then transmit it to the destination entities which know the key. *Public key* systems [3] are familiar to realize the secure communication among two entities. If the secret key is neither hidden nor inferred, *secrecy* and *authenticity* of *secure* communication are held. [14, 15] discuss how to distribute the secret key to only entities which would like to have secure communication by using the public key system and one-to-one communication.

In networks like local area networks (LANs), broadcast communication is supported by the media access control (MAC) layer [7]. In the *group* communication, data units are delivered to multiple entities in the group. A group of entities is named a *cluster* in this paper. In the group communication protocols [11, 12, 13, 16, 17], properties of broadcast communication, i.e. *receipt atomicity* and *receipt ordering* of data units in the cluster are discussed. In this paper, we discuss how to provide the secure broadcast communication for a cluster of multiple entities by using a less-secure broadcast network. We present a protocol to make agreement of the common secure key among only and all the entities in the cluster.

In the teleconferencing, only some members may like to have secure communication in order to have the private talk among them. A *subcluster* is a subset of the cluster. Thus, it is important to provide such secure subcluster communication that some entities other than ones in the subcluster can neither listen to the communication nor send PDUs to the entities in the subcluster. We discuss how to establish a secure subcluster in the cluster.

In section 2, we present a communication model. In section 3, a secure cluster concept is defined. In section 4, a procedure to establish a secure cluster is discussed. In section 5, we discuss how to make a secure subcluster in a cluster.

2 Communication Model

A communication system is composed of three layers, i.e. *application*, *system* and *network* ones. The system layer is composed of system entities [Figure 1]. Each system entity E takes a network service through a network *service access point* (SAP) NS with which E is attached. NS has a unique network address NA. Let $Address(E)$ denote NA. The system layer provides some service for application entities through system SAPs. E supports the system service for application entity A through a system SAP, SS, whose address is SA. Here, let $Name(E)$ denote SA.

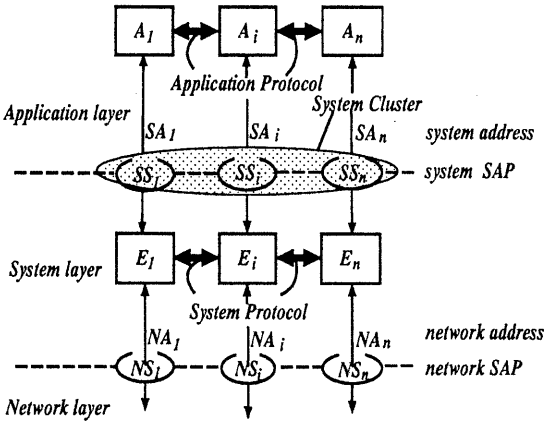

Figure 1: System model

A system *cluster* C is a tuple of system SAPs $\langle SS_1, \ldots, SS_n \rangle (n \geq 2)$. Each SS_j is supported by E_j, i.e. $Name(E_j) = SA_j$ $(j = 1, \ldots, n)$. C is written as $\langle E_1, \ldots, E_n \rangle$. A data unit sent at SS_j in C by each A_j is surely delivered to all the system SAPs in C [11, 12, 13]. A group $\{A_1, \ldots, A_n\}$ can communicate through C where each A_j takes the service at SS_j $(j = 1, \ldots, n)$. C is referred to as *supported* by $E_1, \ldots, E_n$, and *support* $A_1, \ldots, A_n$. $E_1, \ldots, E_n$ are referred to as *included* in C. We assume that all the data units sent at each NS_j are delivered to all the network SAPs in the sending order without any loss and duplicate. In this paper, we would like to discuss how the sys-

tem layer provides secure cluster communication for the application layer by using a reliable but less-secure broadcast communication network.

A data unit exchanged among system entities is named a system *protocol data unit* (PDU). Each PDU p sent by E_j to establish a cluster $C = \langle E_1, \ldots, E_n \rangle$ includes the following attributes.

$p.SRC$	=	system address of SS_j, i.e. $Name(E_j)$.
$p.ADDR$	=	network address of NS_j, i.e. $Address(E_j)$.
$p.DEST$	=	$\{Name(E_i) \mid E_i$ is the destination of p $(i = 1, \ldots, n)\}$.
$p.DATA$	=	data.

p is broadcast to all the network SAPs by using the network service. On receipt of p, each E_j accepts p if $Name(E_j) \in p.DEST$. We make the following assumptions.

[Assumptions] Let p be a system PDU which is sent by E_j.

1. p is delivered to all the network SAPs.
2. $p.SRC$ and $p.DEST$ can be written by E_j.
3. E_j cannot write $p.ADDR$. □

The first assumption means that the network service provides reliable broadcast communication [11, 12, 13, 16, 17]. That is, some system entity E_k $(k \neq j)$ can receive p although E_k is not the destination of p. *Secrecy* may be violated. The second means that the source network address is attached automatically to p at the network layer and system entities cannot change $p.ADDR$. On receipt of p, every E_k understands at which network SAP p is transmitted. The third means that every E_j can write $p.DEST$ and $p.SRC$ when E_j broadcasts p. That is, some E_m $(m \neq j)$ can pretend to be E_j by sending p and using $Name(E_j)$ in $p.SRC$. *Authenticity* may be violated. According to the second assumption, when E_j receives p, although E_j is not sure whether p is broadcast by a system entity denoted by $p.SRC$, it is sure that p is transmitted at $p.ADDR$.

If only and all the system entities $E_1, \ldots, E_n$ in C use the same secret key K to encipher PDUs, C can be secure. K is named a *cluster key* of C. In this paper, we discuss how to distribute K to only and all proper system entities by using the public key system [3] and using the less-secure broadcast service supported by the network layer.

3 Secure Cluster

For every PDU p, let $p.IDATA$ denote data which the source system entity intends to transmit to the destinations. $p.DATA$ may not be the same as $p.IDATA$ if $p.IDATA$ is enciphered to $p.DATA$. A system entity E_j is referred to as *securely receive* p iff E_j receives p and E_j can get $p.IDATA$ from $p.DATA$. Only system entities which know the deciphering key can get $IDATA$. Let us consider the secure communication in a system cluster $C = \langle E_1, \ldots, E_n \rangle$.

[Definition] C is *secret* iff every PDU broadcast is securely received by only entities in C. □

Any entity not in C cannot pretend to be a member in C. Every PDU broadcast by some entity in C is surely received by only entities in C.

[Definition] C is *authentic* iff PDUs broadcast by only entities can be securely received by entities in C. □

Any entity not in C cannot broadcast PDUs to entities in C. Every PDU received by each entity in C is surely one broadcast by some entity in C.

[Definition] C is *secure* iff C is both secret and authentic. □

A secure C can be *established* if only and all the entities in C obtain a common cluster key K. An algorithm by which C is securely established is a *secure cluster establishment* (SCE) procedure. After C is securely established, it is clear that C is secure because PDUs in C are enciphered by using K and only and all the entities in C know K. Problem is how to establish C among multiple entities in the presence of malicious entities. System entities are *malicious* if they are not in C but pretend to be entities in C and listen to communication in C.

Let EE be a set of all the system entities, SA be a set of all the system addresses, and NA be a set of all the network addresses in the system layer. C is a subset of EE. For E in EE, $PName(C, E)$ gives a system address which E uses as its system address in C. That is, when E sends p, E writes $PName(C, E)$ in $p.SRC$. $PName(C, E)$ is not the same as $Name(E)$ if E pretends to be another entity in C.

[Definition] E is *proper* in C iff $Name(E) = PName(C, E)$. E is *malicious* in C iff E is not proper in C. □

If there exists a malicious entity M which pretends to be a proper entity E_j in C, E_j is referred to as *pretended* by M. Note that E_j can receive PDUs broadcast by M pretending E_j, because the underlying network layer provides reliable broadcast communication. Let $Dom(C) = \{ E_j \mid Name(C, E_j) \in C \}$ be a domain of C, which is a set of system entities which try to be a member of C. There are two kinds of entities in $Dom(C)$, i.e. *active* and *passive* ones. *Active* entities require entities to establish C. *Passive* ones wait for the establishment request from the active ones. Let $Active(C)$ and $Passive(C)$ be sets of active and passive entities in C, respectively. Every entity is either *passive* or *active* in C. A *proper domain* of C, $PDom(C)$, is a set of proper system entities in C, i.e. $PDom(C) = \{ E_j \mid Name(E_j) = PName(C, E_j) \}$. $Dom(C)$ may include malicious entities but $PDom(C)$ includes only proper entities.

[Definition] $Dom(C)$ is *secure* iff $Dom(C) = PDom(C)$. $Dom(C)$ is *establishable* iff $Dom(C) \subseteq PDom(C)$ and $Active(C) \cap PDom(C) \neq \phi$. $Dom(C)$ is *unestablishable* iff it is not establishable. □

[Definition] A *secure cluster establishment* (*SCE*) procedure is *complete* iff it can establish a secure cluster on only and all establishable domain. □

If there are all the proper entities in $Dom(C)$ and at least one proper entity is active, i.e. $Active(C) \cap PDom(C) \neq \phi$, the complete *secure cluster establishment* (*SCE*) procedure can securely establish C on $Dom(C)$. If all the active entities are malicious, some complete cluster establishment procedure never establishes some secure cluster. If some proper entity would not like to join C, i.e. $Dom(C) \not\subseteq PDom(C)$, C is not established.

4 Cluster Establishment

We present a complete *secure cluster establishment* (*SCE*) procedure to establish a secure cluster $C = \langle E_1, \ldots, E_n \rangle$.

4.1 Basic procedure

We make the following assumptions on each system entity E_j.

[Assumptions]

1. E_j has a secret key SK_j and a public key PK_j [4].
2. E_j does not know the network address NA_k of every E_k, $Address(E_k)$ $(k \neq j)$.
3. E_j can know a system address $SA_k = Name(E_k)$.
4. E_j can know the public keys of all the system entities.
5. E_j is always operational. □

Here, for each system address SA, let PK_{SA} be a public key of a system entity whose system address is SA. Each entity E_j does not know the network addresses of the other entities. E_j sends a request r to system entities to join C by using their system addresses. In turn, on receipt of r, passive entities know the network address of E_j. Problem is that some malicious entity may use a system address of another proper entity. Hence, in the cluster establishment procedure, each entity has to identify the network addresses of the proper entities.

For every public or private key Y, we assume that $Y(\langle v_1, \ldots, v_n \rangle) = \langle Y(v_1), \ldots, Y(v_n) \rangle$. We assume that every entity has a one-way function F such that for some tuple t of values, $F(t)$ gives some key K. Every E_j has n variables $t_1, \ldots, t_n$, which are initially 0. A *nonce*, e.g. random number of each E_i which E_j knows is stored in t_i $(i = 1, \ldots, n)$.

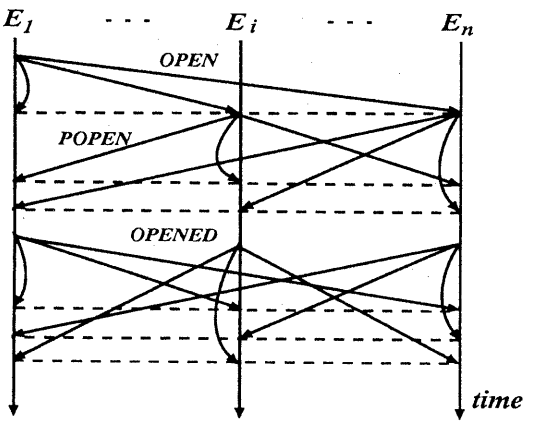

Figure 2: Basic procedure

[Basic Procedure]

1. An active entity E_j broadcasts an *OPEN* PDU p where $p.DATA = SK_j(\langle PK_1(t_j), \ldots, PK_n(t_j)\rangle)$, $p.SRC = T_j$, and $p.DEST = \{T_1, \ldots, T_n\}$ where $T_i = PName(C, E_j)$ $(i = 1, \ldots, n)$, and t_j is a *nonce* of E_j.

2. When an *active* or *passive* entity E_k receives the *OPEN* p from E_j, where $p.DATA = \langle a_1, \ldots, a_n\rangle$, if $Name(E_k) \in p.DEST$, then E_k accepts p and $t_j = SK_k(PK_j(a_k))$. The nonce of E_k is stored in t_k. E_k broadcasts a *POPEN* PDU p where $p.DATA = SK_k(\langle PK_1(t_k), \ldots, PK_n(t_k)\rangle)$. On receipt of the *POPEN* from another entities, E_k executes this step.

3. On receipt of the *OPEN* or *POPEN* PDUs from all the entities, E_k broadcasts an *OPENED* PDU p where $p.DATA = SK_k(\langle PK_1(t), \ldots, PK_n(t)\rangle)$ where $t = \langle t_1, \ldots, t_n\rangle$.

4. On receipt of the *OPENED*s from all the entities, E_k obtains a cluster key $K = F(\langle t_1, \ldots, t_n\rangle)$. Here, C is securely established. □

At the step 4, every entity surely knows that every entity in C has the same tuple of the nonces $\langle t_1, \ldots, t_n\rangle$. However, each proper entity cannot identify malicious ones. Hence, we need a more complicated procedure to be presented in the following subsection.

4.2 Procedure in the presence of attackers

Every system entity E_j has the following variables.

1. t_k = nonce of E_k which E_j receives from E_k $(k = 1, \ldots, n)$.

2. R_{SA_k} = set of tuples $\langle NA, t, type\rangle$ where NA is a network address of E_h whose $PName(C, E_h)$ is $Name(E_k)$, t is the nonce of E_h, and *type* is $A(active)$ if E_h is active, otherwise $P(passive)$ $(k = 1, \ldots, n)$.

3. $TYPE = A$ if E_j is active, otherwise P.

Let R_{SA} be a tuple $\langle R_{SA_1} \ldots, R_{SA_n}\rangle$. Initially, each t_k is 0 and $R_{SA_k} = \phi$ $(k = 1, \ldots, n)$. Each time when E_j receives a nonce t from E_h which uses $Name(E_k)$, $\langle Address(E_h)$, t, A or $P\rangle$ is appended into R_{SA_k}. For a tuple $t = \langle v_1, \ldots, v_n\rangle$, let $t[j]$ denote the j-th element v_j of t.

A *secure cluster establishment* (*SCE*) procedure to establish a secure cluster $C = \langle E_1, \ldots, E_n\rangle$ is presented as follows. Each entity has variables t_j to store the nonces sent from the entities E_j $(j = 1, \ldots, n)$, *TYPE* to store a type of E_i, i.e. *passive* or *active*, and $R = \{R_{SA} \mid SA = Name(E_j) \wedge E_j \in C\}$.

[SCE Procedure]

1. E_j stores its nonce in t_j, 0 in t_k $(k = 1, \ldots, n, k \neq j)$, and A in $TYPE$. E_j broadcasts an *OPEN* PDU p where $p.DATA = SK_j(\langle PK_1(t_j), \ldots, PK_n(t_j)\rangle)$. E_j waits for *OPEN*s or *POPEN*s.

2. [Passive E_k] $TYPE = P$. On receipt of the *OPEN* p where $p.SRC = SA_j$, E_k gets the nonce $u_j = SK_j(PK_k(p.DATA)[j])$. $R_{SA_j} = R_{SA_j} \cup \{\langle p.ADDR, u_j, A\rangle\}$. E_k broadcasts a *POPEN* p where $p.DATA = SK_k(\langle PK_1(t_k), \ldots, PK_n(t_k)\rangle)$ and $p.SRC = SA_j$.

3. [Passive or Active E_h] On receipt of the *OPEN* or *POPEN* p where $p.SRC = SA_j$, if $SA_j \neq Name(E_h)$, E_h gets $u_j = SK_h(PK_j(p.DATA)[h])$. $R_{SA_j} = R_{SA_j} \cup \{\langle p.ADDR, u_j, type\rangle\}$ where $type = A$ if SA_j is active, otherwise P. If $SA_j = Name(E_h)$, then p is neglected.

4. [E_h receives all *OPEN*s or *POPEN*s] On receipt of *OPEN*s or *POPEN*s from all the entities in C, E_h broadcasts an *OPENED* p where $p.DATA = SK_h(\langle PK_1(R_{SA}), \ldots, PK_n(R_{SA})\rangle)$ after waiting for some time.

5. E_h receives an *OPENED* p where $p.SRC$ is SA_j and $p.ADDR$ is NA_j, and $RR_{SA_j} = SK_h(PK_{SA_j}(p.DATA)[h])[j]$ $(j = 1, \ldots, n)$.

   (1) If RR_{SA_j} does not include $\langle Name(E_h), t_h, TYPE\rangle$, E_h neglects p and removes from $RR_{SA_1}, \ldots, RR_{SA_n}$ every tuple whose system address is $p.ADDR$, because the entity denoted by $p.ADDR$ is not proper.

   (2) If $R_{SA_j} \cap R_{SA_j} \neq \phi$, p is neglected. For every i, $R_{SA_i} = R_{SA_i} \bigcap RR_{SA_i}$. If some R_{SA_i} is empty, E_h aborts C.

6. On receipt of all the *OPENED* PDUs, every R_{SA_j} is a singleton in E_h $(j = 1, \ldots, n)$. If

all the tuples in R_{SA} are of *passive* type, E_h aborts C. Otherwise, let K be $F(\langle t_1, \ldots, t_n \rangle)$ where t_j is in R_{SA_j} $(j = 1,\ldots,n)$. □

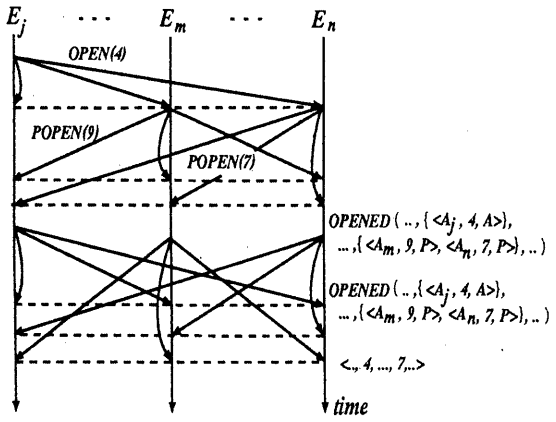

Figure 3: Passive attacker E_m

4.3 Completeness

We show that the *SCE* procedure is complete for C. First, suppose that there is some malicious passive entity E_m who pretends to be a proper E_j. Some active E_a broadcasts an *OPEN* PDU p with the nonce t_a. Since E_j knows the secret key SK_j, E_j can get t_a from p. However, E_m cannot get it. E_j broadcasts t_j, and E_m broadcasts t_m. Another proper E_k receives t_j and t_m from E_j and E_m, respectively. It cannot decide which entity E_j or E_m is proper. RR_{SA_j} includes $\{\langle NA_j, t_j, Passive\rangle, \langle NA_m, t_m, Passive\rangle\}$. After receiving the *OPEN* or *POPEN* PDUs from all the entities who have system addresses in C, every entity broadcasts an *OPENED* PDU. E_k receives the *OPEN* PDU p from E_m. Since RR_{SA_j} carried by p includes t_k, E_k finds that E_m whose system address is $p.ADDR$ is malicious. Thus, malicious passive entities can never join C.

Figure 3 shows an example of passive attack by malicious entity E_m pretending to be E_n. E_j broadcasts an *OPEN* PDU p with a nonce 4 in order to establish a secure cluster C. The proper entities in C can get the nonce 4 by deciphering p by their secret keys. They broadcast a *POPEN* with their nonces, e.g. E_n sends the nonce 7. E_m broadcasts a *POPEN* q with some nonce on receipt of p. Suppose that proper entities, e.g. E_j and E_n get a nonce 9 from q. Since E_j and E_n have two nonces 7 and 9 for $Name(E_n)$, they broadcast *OPENED*s with $\langle\langle A_m, 9, P\rangle, \langle A_n, 7, P\rangle\rangle$. Since E_m cannot get the nonce 4 for E_j, E_m broadcasts an *OPENED* which has a different tuple of nonces from the other entities, the proper entities know that E_m is malicious and get the same tuple of nonces $\langle..,4, \ldots, 7,..\rangle$. Then, only and all the proper entities in C get the same cluster key from the same tuple of nonces.

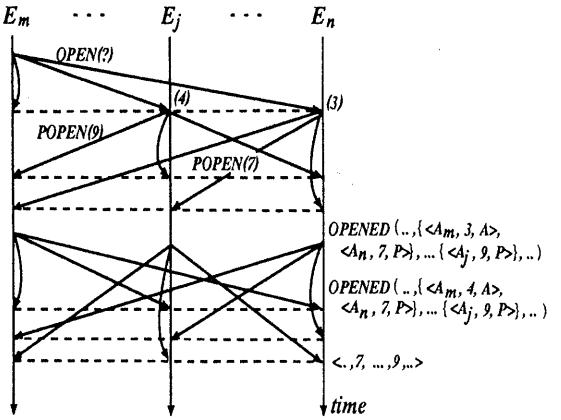

Figure 4: Active attacker E_m

Next, suppose that there exists some active attacker E_m which pretends to be a proper E_k. First, E_m broadcasts an *OPEN* p. Every E_j receives p and obtains some nonce tt_k by deciphering $p.DATA$. Since E_m broadcasts an arbitrary nonce, each entity may obtain different nonce for $Name(E_k)$. Since E_m does not know the secret key of E_k, E_m does not know the nonce which E_j obtains from the PDU sent by E_m. E_j and E_k broadcast t_j and t_k, respectively. Thus, E_j receives two nonces tt_k and t_k for E_m and E_k for $Name(E_k)$, respectively, on receipt of the *OPEN* or *POPEN* PDUs from all the entities in C, every entity broadcasts the *OPENED*. If E_m broadcasts an *OPENED* p, every E_j finds that E_m is malicious because E_m does not send back the same nonce sent by E_j. Every entity agrees that E_k is proper. If E_m broadcasts no PDU, every entity times out and finds that E_m is malicious. In both cases, E_m cannot obtain the cluster key K since E_m cannot know any nonce of the proper entities. Thus, every proper E_j $(j = 1, \ldots, n)$ can find which entity E_m or E_k is proper. Here, if all the active entities are malicious, the cluster is not established by the step 6 of the *SCE* procedure. C cannot be established unless there is at least one active proper entity. This means that C is

never maliciously established where no entity in the cluster want to join it.

In Figure 4, a malicious entity E_m pretends to be E_n. E_m broadcasts an *OPEN* PDU p with some nonce t. On receipt of p, proper entities get the nonce by deciphering p. Suppose that E_j gets 4 and E_n gets 3 as the nonce of $Name(E_n)$. Proper entities broadcast *POPEN* PDUs with their nonces, e.g. E_j and E_n send the nonces 9 and 7, respectively. After broadcasting *OPEN* and *POPEN* PDUs, E_j broadcasts a tuple including $\{\langle A_m, 4, A\rangle, \langle A_n, 7, P\rangle\}$, and E_h broadcasts $\{\langle A_m, 3, A\rangle, \langle A_n, 7, P\rangle\}$. So, the proper entities find that E_m is malicious. Since E_m is only one active entity, the cluster is not opened.

As stated above, it is clear for the following theorem to hold.

[**Theorem**] The *SCE* procedure is complete. □

4.4 Performance

First, we consider the PDU complexity. In the *SCE* procedure, every entity broadcasts two PDUs, i.e. either *OPEN* or *POPEN*, and *OPENED* PDUs. Let n be the number of proper entities in C, and $m(\leq n)$ be the number of malicious entities. The maximum number of PDUs broadcast to establish C is $2 \times (n + m)$, i.e. all the malicious entities try to join C by pretending to be some proper entities. On the other hand, the minimum number is $2 \times n$ i.e. there is no malicious entity.

Next, let us consider the time complexity. Here, we define a *round* to be a maximum delay time to deliver a PDU from one network SAP to all the network SAPs. In the best case, the cluster can be established by 3 rounds when *POPEN* and *OPENED* PDUs are broadcast in parallel. At worst, it takes $2 \times (n+m)$ rounds where at most one PDU are broadcast simultaneously in the network layer.

The length of the *OPEN* and *POPEN* PDU is $O(n)$. The length of the *OPENED* PDU is $O((n+m) \times n)$.

5 Secure Subcluster

In the group communication like *tele-conferencing* [6], some members of a cluster $C = \langle E_1, \ldots, E_n\rangle$would rather have more secure communication while having secure cluster communication in C. For example, it is usual that we would like to have a private talk with our neighbors in a meeting. Suppose that only $E_{s_1}, \ldots, E_{s_m}$ $(m \leq n)$ in C would like to have secure *private* communication. There are two ways to realize such secure communication. One way is to establish a secure cluster D for $E_{s_1}, \ldots, E_{s_m}$ independently of C by using the *SCE* procedure. The other way is to establish a secure *subcluster* C_h for $E_{s_1}, \ldots, E_{s_m}$ in C. In this case, E_h is established by using the secure cluster communication provided by C. Although every PDU p broadcast by some E_{s_i} can be delivered to all the entities in C, p can be understood by only entities $E_{s_1}, \ldots, E_{s_m}$ in C_h. C_h is securely established by exchanging the nonces enciphered by the cluster key K in the same procedure as the *SCE* procedure. In result, only and all $E_{s_1}, \ldots, E_{s_m}$ can obtain the common secret key K_h for C_h. If each E_{s_i} would like to send a PDU p to only the members of C_h, E_{s_i} enciphers p by K_h. In the former approach, the cluster D for $E_{s_1}, \ldots, E_{s_m}$ has to be established for all the system entities. On the other hand, in the latter *subcluster* approach, the subcluster C_h is established for C. In this sense, the subcluster approach has less possibility of having malicious attacks and requires less PDUs to be broadcast to establish C_h than the former one.

6 Concluding Remarks

In this paper, we have presented a protocol which supports the secure group communication among multiple entities by using a less-secure broadcast communication service like the Ethernet. In the broadcast network, every entity can receive PDUs broadcast by every entity. Malicious entities can receive PDUs broadcast and can broadcast PDUs as another entity. In the presence of these attacks, we have shown a complete *secure cluster establishment* (*SCE*) procedure which can establish a secure cluster for an *establishable* domain based on the public key system. In addition, we have discussed how to establish a secure subcluster in the cluster. At present, we are implementing the secure cluster on the top of the reliable broadcast communication system [11, 12, 13, 16, 17].

References

[1] Birman, K. P., Joseph, T. A., "Reliable Communication in the Presence of Failures,"

ACM Trans. Computer Systems, Vol.5, No.1, 1987, pp.47-76.

[2] Burrows, M., Abadi, M., and Needham, R.M., "Authentication: Practical Study in Belief and Action," *Proc. of the Second Conf. on Theoretical Aspects of Reasoning about Knowledge* (Vardi, M., ed.), 1987, pp.325-342

[3] Denning, D. E., "Cryptography and Data Security," *Addison-Wesley*, 1983.

[4] Diffie, W. and Hellman, M., "New Direction in Cryptography," *IEEE Trans. on Inf. Theory*, Vol. IT-22, No.6, 1976. pp.644-654

[5] Defense Communications Agency, "DDN Protocol Handbook," Vol.1 - 3, NIC 50004-50005, 1985.

[6] Ellis, C. A., Gibbs, S. J., and Rein, G. L., "Groupware," *CACM*, No.1, 1991, pp.38-58.

[7] "IEEE Project 802 Local Network Standards-Draft," 1982.

[8] ISO, "Basic Reference Model," *IS* 7498, No.1-3, 1986.

[9] Kaashoek, M. F. and Tanenbaum, A. S., "Group Communication in the Amoeba Distributed Operating System," *Proc. of the 11th IEEE ICDCS*, 1991, pp.222-230.

[10] Lamport, R., "Time, Clocks, and the Ordering of Events in Distributed Systems," *CACM*, Vol.21, No.7, 1978, pp.558-565.

[11] Nakamura, A. and Takizawa, M., "Reliable Broadcast Protocol for Selectively Ordering PDUs," *Proc. of the 11th IEEE ICDCS*, 1991, pp.239-246.

[12] Nakamura, A. and Takizawa, M., "Design of Reliable Broadcast Communication Protocol for Selectively Partially Ordered PDUs," *Proc. of the 11th IEEE COMPSAC'91*, 1991, pp.673-679.

[13] Nakamura, A. and Takizawa, M., "Priority-Based Total and Semi-Total Ordering Broadcast Protocols," *Proc. of the 12th IEEE ICDCS*, 1992, pp.178-185.

[14] Needham, R.M. and Schroeder, M.D., "Using Encryption for Authentication in Large Networks of Computers," *CACM*, Vol.21, No.12, 1978, pp.993-999.

[15] Needham, R.M. and Schroeder, M.D., "Authentication Revisited," *ACM Operating Systems Review*, Vol.21, No.1, 1987, p.7.

[16] Takizawa, M., "Cluster Control Protocol for Highly Reliable Broadcast Protocol," *Proc. of the IFIP Conf. on Distributed Processing*, 1987, pp.431-445.

[17] Takizawa, M., "Design of Highly Reliable Broadcast Communication Protocol," *Proc. of the 11th IEEE COMPSAC'87*, 1987, pp.731-740.

[18] Takizawa, M. and Nakamura, A., "Totally Ordering Broadcast (TO) Protocol on the Ethernet," *Proc. of the IEEE Pacific RIM Conf. on Communications, Computers and Signal Processing*, 1989, pp.16-21.

[19] Takizawa, M. and Nakamura, A., "Partially Ordering Broadcast (PO) Protocol," *Proc. of the 9th IEEE INFOCOM'90*, 1990, pp.357-364.

[20] Takizawa, M. and Nakamura, A., "Reliable Broadcast Communication," *Proc. of IPSJ Int'l Conf. on Information Technology (InfoJapan)*, 1990, pp.325-332.

[21] Tanenbaum, A. S,: "Computer Networks," *Prentice-Hall Int'l, Inc*(1989).