공모공격을 차단하는 효율적인 그룹 키 관리

김 태 균[†] · 정 종 인[††]

요 약

인터넷상에서 멀티캐스트 서비스가 급속도로 증가하고 있으며 이에 따라 멀티캐스트 통신의 보안을 유지하는 것이 중요하다. 멤버의 탈퇴는 그룹키 관리의 확장성 문제와 밀접한 관계가 있다. 그룹의 한 멤버가 제거되면 새로운 그룹키를 생성하여 그룹의 나머지 모든 멤버들에게 전달되어야 한다. 새로운 키를 생성하여 분배하는 것은 많은 연산을 요구하므로 rekey하기 위하여 보내는 메시지의 수와 복합키를 생성하기 위한 연산비용을 최소화하는 것은 키 관리기법을 평가하는 중요한 기준이다. 일괄제거는 멤버를 순차적으로 1개씩 제거하는 것보다 rekey에 대한 메시지의 수와 복합키의 연산 비용을 줄일 수 있다. 본 논문에서는 그룹에서 제거될 멤버들간의 해밍거리가 임계치보다 작은 멤버들만 동시에 제거된다. 여러 개의 멤버를 제거할 때 라운드 조정 알고리즘을 수행하면 rekey를 위하여 메시지의 수와 복합키를 생성하기 위한 연산의 비용을 줄이며 공모공격의 가능성이 제거되는 이점이 있다.

The Efficient Group Key Management Blocking Collusion Attack

Tae-Gyun Kim[†] · Jong-In Chung[††]

ABSTRACT

Multicast services are provided on the Internet in fast increasing. Therefore it is important to keep security for multicast communication. Member leaving is deeply associated with scalability problem for group key management. If one member of the group is removed, new group key has to be generated and distributed to all remaining members of group. Minimizing the number of messages and operation cost for generation of the composite keys are important evaluating criteria of multicast key management scheme since generation and distribution of new keys for rekeying require expensive operation. Batch removal can reduce these important parameters rather than removing members sequentially in fashion one after another. In this paper, Hamming distance is calculated between every members to be removed. The members with Hamming distance less than threshold are selected for rekeying procedure. With running the round assignment algorithm in the case of removing several members simultaneously, our model has advantages of reducing the number of message and operation cost for generation of the composite keys and eliminating possibility of collusion attack for rekeying.

키워드 : 그룹키(Group Key), 키관리(Key Management), 공모공격(Collusion Attack)

1. 서 론

멀티캐스트 서비스는 1개 이상의 발신자로부터 인증된 많은 수신자들에게 메시지를 보낸다. 수신자는 IGMP 메시지를 로컬 라우터에게 보냄으로서 D급 클래스인 멀티캐스트 그룹에 가입하고 또한 탈퇴할 수 있다. 인터넷에서 멀티캐스트는 큰 그룹에 효율적이고 최선의 노력으로 전달서비스를 제공하고 있다. 그러므로 멀티캐스트 통신의 보안을 유지하는 것이 중요하다. 즉, 그룹 멤버들 사이 전달되는 메시지가 무결성, 기밀성, 확장성(scalability)이 제공되어야 한다[1-3]. 멀티캐스트 서비스를 하기 위하여 유니캐스트를 반복하여 사용할 수 있으나 확장성을 제공하기에는 한계가 있다.

수신자의 수를 제한하거나 데이터의 출처를 인증하여야 하는 응용에서는 그룹의 인증이 지원되어야 한다. 디지털 미디어의 VOD서비스, 온라인 게임, 원격회의와 같은 응용은 정상적인 가입자들만이 수신할 수 있도록 가입자를 인증한다. 더불어 주식정보를 분배하는 응용은 데이터의 출처를 인증한다.

멀티캐스트 그룹은 키 서버에 의해 관리된다. 이러한 키 서버를 그룹제어기라 한다. 멀티캐스트 그룹에 가입하기 위하여 클라이언트는 그룹제어기에게 그룹에 접근을 요구하여야 한다. 그룹제어기는 요구를 받으면 클라이언트의 로그인 명과 패스워드나 증명서에 의해 신원을 확인한다. 클라이언트가 그룹에 가입이 허용되면 제어기는 메시지가 전달될 멀티캐스트 주소와 필수적인 키를 클라이언트에게 제공한다.

※ 본 연구는 과학기술부 목적기초연구(과제번호:R05-2004-000-11027-0) 지원으로 수행되었음.
† 준 회 원 : 공주대학교 대학원 컴퓨터교육과
†† 정 회 원 : 공주대학교 컴퓨터교육과 교수
논문접수 : 2004년 1월 29일, 심사완료 : 2004년 6월 16일

클라이언트에게 그룹의 모든 멤버들에 의해 공유되는 그룹키와 키분배 알고리즘에서 사용되는 보조키를 분배한다.

제어기는 또한 멤버의 가입과 탈퇴를 담당한다. 멤버의 가입과 탈퇴는 그룹키 관리의 확장성문제와 관계가 있다. 1개의 그룹키를 공유하며 N개의 멤버로 구성되는 멀티캐스트 그룹을 생각하여 보자. 그룹에 한 멤버가 가입하면 그 가입자에게만 그룹키를 암호화하여 안전한 채널을 사용하여 유니캐스트 통신으로 전달하면 된다. 멤버 가입은 탈퇴보다 그룹키 전달 비용이 적게 든다. 그룹의 한 멤버가 탈퇴한다면, 그룹키를 변경하여 그룹의 나머지 $N-1$개의 멤버에게 전달되어야 한다. 그러나 새로운 그룹키를 나머지 모든 멤버들에게 안전하게 보내는 것은 간단한 문제가 아니다. 아주 간단한 해결책은 제어기와 그룹의 나머지 멤버들간에 유일한 키를 가지고 유니캐스트 통신을 하는 것이다. 이 방법은 간단하지만 $N-1$개의 유니캐스트 연결과 N개의 비밀키를 요구하기 때문에 확장성이 좋지 않다.

Chang[4], Wong[5]는 확장성을 제공하기 위하여 계층구조의 그룹키를 갖는 관리 체제를 제안하였다. 그룹키 변경시 Wong은 키 관리 트리의 어떤 키를 그의 자식노드의 키에 의해 암호화되어 멀티캐스트하는 전형적인 LKH(Logical Key Hierachy) 모델를 제안하였으며 그룹의 크기가 N개이면 제어기가 1개의 그룹키, $N-2$개의 보조키, N개의 개인키를 관리하여야 한다.

Chang은 멤버들에게 2진수 ID를 할당하며, 키 트리의 각 노드의 키는 2진수 ID에 의해 결정되므로 키의 관리를 용이하며 LKH 키관리 방식을 사용하였다. 그룹의 크기가 N개이면 제어기가 1개의 그룹키, $2\log N$개의 보조키를 관리하여야 한다.

Chang과 Wong모델에서는 1개의 멤버를 그룹으로부터 제거하기 위하여 제어기가 $\log N$개와 $2\log N-1$개의 메시지를 각각 멤버들에게 보낸다. 그룹의 크기가 크고 가입과 탈퇴가 빈번할 경우, 각 멤버를 제거할 때마다 새로운 그룹키를 갱신하고 분배하는 것은 많은 연산을 요구한다[6, 7]. Chang은 탈퇴자의 수가 적정선에 도달하거나 주기적인 시간마다 새로운 키를 생성하는 방안을 제안하였으나 탈퇴 멤버들이 공모하여 키의 보안성을 파괴하는 공모공격의 가능성이 있다.

본 논문에서는 Chang의 키 관리체제를 사용하지만, 제거대상 멤버들의 제거 라운드를 조정함으로서 Chang모델보다 메시지의 수와 메시지를 암호화하는데 사용하는 복합키의 연산 비용을 줄이며 공모공격(collusion attack)의 문제점을 해결한다.

2. 키 관리 체제

그룹의 각 멤버에는 n비트 길이의 2진수 문자열을 가진 사용자 ID(UID)가 부여된다. UID는 $x_{n-1}x_{n-2}\cdots x_0$로 표현되어지며 각 x_i는 0이나 1이다. UID의 길이는 그룹 크기에 의존한다. 예를 들면 그룹의 크기가 300이면 9비트 UID를 사용한다.

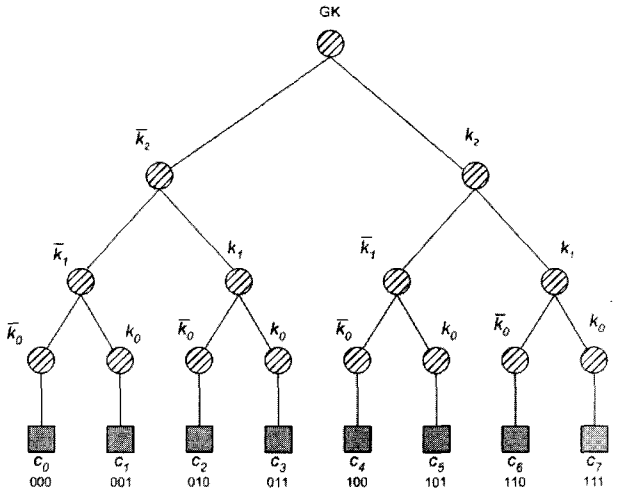
UID $x_{n-1}x_{n-2}\cdots x_0$인 멤버가 그룹에 가입하기 위하여 그룹 제어기에게 등록을 하면 제어기는 멤버에게 공용 그룹키 GK를 보낸다. GK는 그룹의 모든 멤버들이 공유하며, 멀티캐스트 그룹에게 메시지를 보낼 때 암호화하거나 복호화할 때 사용된다. 또한 멤버는 n개의 보조키 세트 K_{n-1}, K_{n-2}, …, K_0를 받는다. 여기서 K_i는 x_i가 1이면 k_i, 0이면 $\overline{k}_i$로 표현된다. 보조키는 그룹키를 안전한 방법으로 갱신하는데 사용된다. k_i와 $\overline{k}_i$는 서로 보수키(complement key)이며, 수치적인 보수관계가 아니라 서로 관계가 없는 키라는 의미이다. 제어기는 모든 보조키 $\{k_0, \overline{k}_0, k_1, \overline{k}_1, \cdots, k_{n-1}, \overline{k}_{n-1}\}$를 관리한다.

(그림 1)은 그룹의 크기가 8인 경우에 각 멤버가 보관하는 키를 보여주고 있다. 트리에서 사각형 단말 노드는 3비트의 UID를 가지는 그룹 멤버를 나타내고, 트리의 원형 노드는 키를 나타낸다. 각 멤버는 단말 노드에서 루트 노드까지 가는 가지(branch)에 있는 키를 보관한다. 예를 들면, 멤버 c_5(UID 101)은 그룹키 GK와 보조키 k_2, $\overline{k}_1$, k_0를 보관한다.

일반적으로 그룹키와 보조키는 그룹 멤버가 탈퇴할 때 변경된다. 그룹을 탈퇴한 멤버가 미래에 사용될 키들을 획득할 수 없어야 한다. 그러기 위해서는 그룹 멤버가 탈퇴하면 다른 멤버가 가지고 있는 그룹키와 보조키를 변경함으로서 탈퇴한 멤버가 가지고 있는 보조키 및 그룹키를 사용하여 미래의 키들을 획득할 수 없게 한다. 이와 같이 키의 변경을 rekey라 한다. 그룹멤버가 탈퇴할 때마다 rekey를 하면 그룹키 및 보조키를 변경하여야 하므로 많은 비용이 들게 된다. 또한 2개의 멤버가 짧은 시간내에 연속으로 탈퇴할 때 첫 번째 멤버에 대한 rekey에 의해 생성된 새로운 그룹키와 보조키를 사용하지도 못하고 두 번째 멤버에 대한 rekey를 하여야 한다. 이것은 자원의 낭비를 초래하게 된다. 그 외에 멤버의 탈퇴가 있을 때마다 rekey하면 키와 데이터간의 동기화가 안되는 문제(out-of-synchronization)가 발생한다[8]. 대부분의 응용에서는 멤버가입과 탈퇴 요구가 있을 때 즉시 처리할 필요가 없다. 예를 들면 VOD와 온라인 게임에서 그룹키 관리를 위하여 사용자가 지불한 요금에 해당하는 서비스 시간이 경과되자마자 바로 탈퇴시키지 않아도 된다.

그룹 제어기가 일정한 주기마다 그룹에서 탈퇴할 멤버들을 모아서 한꺼번에 rekey하는 것을 일괄(batch) rekey라 한다. 그리하여 본 연구에서는 일괄 rekey방식을 사용한다. 제거할 멤버를 모아서 주기적으로 제거하는 것을 일괄제거(batch removal)라 하고 이러한 주기를 라운드(round)라 한

다. 라운드의 주기는 제어기가 임의로 정할 수 있다. Rekey는 매 주기마다 불연속적으로 일어나므로 그룹키와 보조키를 각각 $GK(r)$과 $k_i(r)$, $\overline{k}_i(r)$로 표현한다. 여기서 r은 현재의 라운드를 나타낸다.

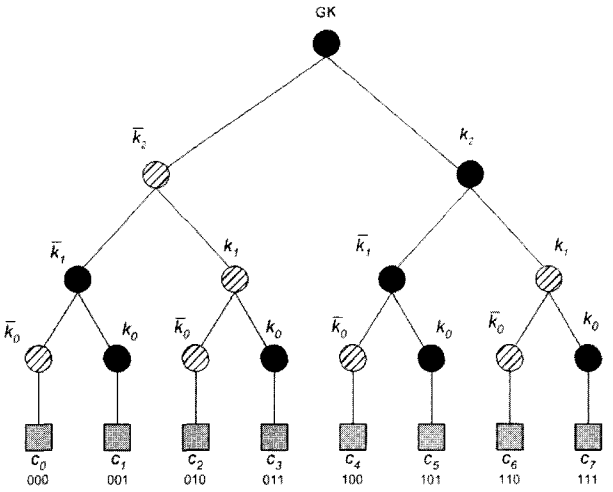


(그림 1) 그룹크기 8의 키 분배트리

2.1 한 개의 멤버 제거

그룹의 한 멤버가 탈퇴할 때마다 새로운 그룹키가 그룹을 탈퇴하는 멤버를 제외한 모든 멤버들에게 분배되어야 한다. 현재의 그룹키 $GK(r)$를 갱신하기 위하여 제어기는 새로운 그룹키 $GK(r+1)$를 계산한다. $GK(r+1)$는 제거되는 멤버의 보수키를 사용하여 암호화된다. 예를 들면, 제거되는 멤버의 UID가 101이라면 그 멤버는 보조키 k_2, $\overline{k}_1$, k_0을 가지고 있다. 그러므로 $GK(r+1)$는 그 멤버의 보수키 $\overline{k}_2$, k_1, $\overline{k}_0$에 의해 각각 암호화된 3개의 메시지 $\{GK(r+1)\}_{\overline{k}_0}$, $\{GK(r+1)\}_{k_1}$, $\{GK(r+1)\}_{\overline{k}_2}$를 만들어 그룹의 모든 멤버들에게 분배한다. 여기서 $\{L\}_M$은 문자열 L을 키 M에 의해 암호화하여 그룹 전체 멤버들에게 보낸다는 의미이다. 제거될 멤버도 암호화된 모든 메시지를 수신하지만 그 메시지들은 그 멤버가 가지고 있지 않은 키로 암호화되어 있기 때문에 복호화할 수 없다. 그러나 그룹의 나머지 멤버들의 UID는 제거될 멤버의 UID와 적어도 1비트 이상 다르므로 그 멤버들은 암호화된 메시지 중의 적어도 1개 이상을 복호화할 수 있다.

(그림 2)는 멤버 c_5가 제거될 때 그룹 rekey 체제를 나타낸다. 짙은 원형노드는 그룹에서 제거되는 멤버 c_5가 가지고 있는 보조키를 나타낸다. 빗금친 원형노드는 c_5가 가지고 있지 않는 보조키 즉, c_5의 보수키를 나타낸다. c_5에서 루트까지 가는 가지의 모든 노드는 짙은 원형 노드이지만 그 외의 노드에서 루트까지 가는 가지에는 적어도 1개 이상의 빗금친 노드가 존재한다. 그러므로 c_5를 제외한 모든 멤버는 c_5가 가지고 있지 않은 보조키에 의해 암호화된 1개 이상의 메시지를 복호화할 수 있다.



(그림 2) 1개의 멤버 c_5가 제거되는 예

키 분배 알고리즘을 분석하면, 그룹 멤버수가 N인 경우, 제어기에 의해 관리되는 키의 수는 $2\log N+1$개이며, 1개의 멤버가 제거된 후에 그룹키를 갱신하기 위하여 보내는 메시지는 $\log N$개이다.

제거되는 멤버는 새로운 그룹키를 획득할 수 없다. 그룹키를 다음 라운드에서 갱신할 경우 제거되는 멤버가 이 그룹키를 복호화하지 못하도록 보조키를 갱신하여야 한다. 각 멤버는 보조키 $K_i(r)$를 갱신하기 위하여 식 (1)과 같이 해쉬함수 f를 사용한다.

$$K_i(r+1) = f(K_i(r), GK(r+1)) \quad (1)$$

새로운 그룹키 $GK(r+1)$를 가지고 있는 멤버만이 보조키 $K_i(r+1)$를 갱신할 수 있다. 제거되는 멤버는 $GK(r+1)$를 가지고 있지 않기 때문에 새로운 보조키를 갱신할 수 없으며 다음 라운드에서 변경될 그룹키, $GK(r+2)$를 갱신할 수 없으므로 보안이 유지된다.

2.2 여러 개의 멤버 제거

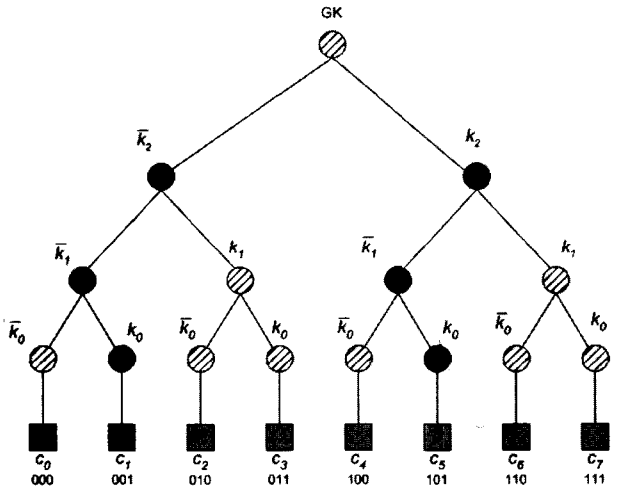
k개의 멤버를 제거하기 위하여 2.1절의 키 갱신 과정을 k번 반복할 수 있다. 그러나 더욱 효과적인 방법은 제거할 멤버들을 모아서 한꺼번에 제거하는 것이다.

일반적으로 멤버의 집합, $S=\{c_0, c_1, \cdots, c_{N-1}\}$이며 $N=2^n$이다. 멤버의 UID를 $x_{n-1}x_{n-2}\cdots x_0$로 표현한다. 어떤 시점에서 그룹의 멤버쉽은 UID의 부울함수 $mem(\,)$에 의해 결정된다. 즉, $mem(x_{n-1}x_{n-2}\cdots x_0)=1$이라면 $x_{n-1}x_{n-2}\cdots x_0$는 그룹의 멤버이고, 그렇지 않으면 그룹에서 제거된다.

그룹 rekey는 $mem(\mathrm{UID})=0$인 멤버를 제외한 모든 멤버에 대하여 그룹키와 보조키를 갱신하는 것이다. 그룹키는

제거될 멤버가 가지고 있지 않는 키에 의해 암호화되어 모든 멤버들에게 전송된다. 확장성과 효율성을 고려하여 rekey는 그룹에 전송될 최소의 메시지와 암호화하는데 최소의 연산이 소요되는 것이 바람직하다.

예를 들면, (그림 3)에서 UID 001(c_1)과 101(c_5)가 그룹으로부터 제거된다고 가정하자. 그룹의 나머지 멤버의 집합 $S=\{c_0, c_2, c_3, c_4, c_6, c_7\}$에게 새로운 그룹키, $GK(r+1)$을 분배하여야 한다. 일반적으로 모든 UID에게 새로운 그룹키가 할당되는 것이 아니므로, 실제 그룹키를 분배하여야 되는 멤버는 S의 부분집합이다. UID가 할당되지 않은 멤버는 키 분배에 영향을 주지 않는 "don't care" 조건으로 취급할 수 있다. 암호화된 그룹키, $\{GK(r+1)\}_{\overline{k}_0}$와 $\{GK(r+1)\}_{k_1}$를 갖는 2개의 메시지를 분배하면 된다. 첫 번째 메시지는 멤버 c_0, c_2, c_4, c_6에 의해 복호화할 수 있으며, 두 번째 메시지는 멤버 c_2, c_3, c_6, c_7에 의해 복호화할 수 있다. 그리하여 2개의 메시지는 그룹 S에서 c_1, c_5를 제외한 그룹을 커버하므로 그룹 rekey가 가능하다. (그림 3)에서 짙은 원형노드는 c_1, c_5가 가지고 있는 키를 나타내며, 이들은 $GK(r+1)$을 암호화하는데 사용할 수 없다.



(그림 3) 2개의 멤버 c_1, c_5가 제거되는 예

1개의 멤버를 제거하기 위한 rekey는 3개의 메시지가 필요하였다. 1개의 멤버를 제거하고 나서 다음 다른 멤버를 제거하는 식의 순서적으로 rekey가 이루어진다면, 2개의 멤버가 제거될 경우에는 $2\times3=6$개의 메시지를 분배하여야 한다. 그러나 2개의 멤버를 모아서 일괄제거할 경우에는 단지 2개의 메시지만 분배하면 된다. 이 수치는 1개의 멤버를 제거할 경우보다 더 적은 수치이다. 직관적으로 그룹의 나머지 멤버들의 UID가 공통인 비트가 1개 이상(공통인 비트는 제거되는 UID의 비트와 다름)이면 분배될 메시지의 수는 줄어들게 됨을 알 수 있다. 위의 예에서 c_0, c_2, c_4, c_6의 UID의 x_0은 제거되는 멤버(c_1, c_5)의 UID의 x_0와 달리 모두 0이다. 제거되는 멤버는 k_0를 가지고 있는 반면에 c_0, c_2, c_4, c_6는 $\overline{k}_0$를 가지고 있기 때문에 $\{GK(r+1)\}_{\overline{k}_0}$를 복호화할 수 있다. 유사하게 c_2, c_3, c_6, c_7의 x_1은 제거되는 멤버(c_1, c_5)의 UID의 x_1와 달리 모두 1이다. 제거되는 멤버는 $\overline{k}_1$를 가지고 있는 반면에 c_2, c_3, c_6, c_7은 k_1을 가지고 있기 때문에 $\{GK(r+1)\}_{k_1}$을 복호화할 수 있다. 동시에 제거되는 멤버들의 UID의 공통비트가 많을수록 그룹의 나머지 멤버들의 공통비트가 많아지므로 rekey를 위한 메시지의 수가 적어진다.

일괄제거 문제는 제거되는 멤버를 제외한 나머지 멤버들의 UID의 비트가 서로 공통인 멤버들의 그룹을 체계적으로 찾는 것과 같은 문제이다. 이러한 문제는 부울 함수의 최소화[9]와 같은 문제이다. 예를 들면, 멤버쉽 함수는 다음과 같이 표현할 수 있다.

$$mem(x_2, x_1, x_0) = \overline{x}_2\,\overline{x}_1\,\overline{x}_0 + \overline{x}_2 x_1\,\overline{x}_0 + \overline{x}_2\,x_1\,x_0 + x_2\,\overline{x}_1\,\overline{x}_0 + x_2 x_1\,\overline{x}_0 + x_2\,x_1\,x_0 \quad (2)$$

여기서 +는 논리적인 OR이며 변수들의 곱은 논리적인 AND를 나타낸다. 식 (2)의 각 항은 각 멤버에게 전달될 그룹키를 암호화한 1개의 메시지에 대응된다. 각 항을 구성하고 있는 키들을 입력으로 하는 일방향 함수에 의해 생성된 복합키로 암호화하여 1개의 메시지를 생성한다. 예를 들면, $x_2\,\overline{x}_1\,\overline{x}_0$항은 $k_2, \overline{k}_1, \overline{k}_0$로부터 생성되는 복합키에 의해 암호화되는 메시지에 대응된다. 그러므로 그룹의 모든 멤버들에게 새로운 그룹키를 전달하기 위하여 6개의 메시지를 분배하여야 한다. 이러한 각 메시지들은 그룹에 남아 있는 6개의 멤버들 중의 1개에 의해 복호화된다.

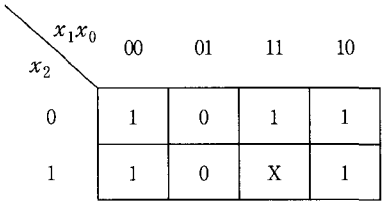
부울함수의 최소화방법이 최소의 메시지와 최소의 키를 가지는 키 갱신 문제에 어떻게 적용되는지를 이해하기 위하여 예를 들어 설명한다. c_7이 그룹을 이미 떠났다고 가정하고, c_7을 제외한 나머지 7개의 멤버들에게 UID가 할당되어 있다고 가정하자. 그룹으로부터 c_1과 c_5를 제거한다면 멤버쉽 함수는 <표 1>과 같다. c_1과 c_5의 출력은 0, 그룹을 이미 떠난 c_7 은 X(don't care), 나머지 멤버들의 출력은 1이다. c_7은 그룹을 떠난 후 보조키를 갱신시켰기 때문에 새로운 키에 의해 암호화된 메시지를 복호화할 수 없다. 멤버쉽 함수를 최소항의 합(sum of minterms)으로 나타내면 식 (3)과 같이 표현할 수 있다. 식 (3)의 Σ는 항들이 OR된 것을 나타내고, 소문자 m과 숫자는 부울함수의 최소항을 나타낸다. 예를 들면, 최소항 $\overline{x}_2\,x_1\,\overline{x}_0$를 $m(2)$, $x_2 x_1\,\overline{x}_0$를 $m(6)$으로 표현한다. 또한 $d(7)$은 UID가 7인 don't care항을 나타낸다.

$$mem(x_2, x_1, x_0) = \sum m(0, 2, 3, 4, 6) + d(7) \quad (3)$$

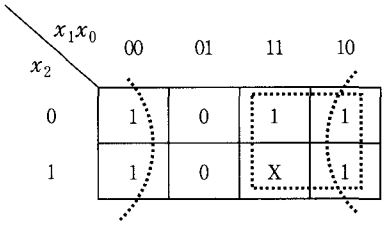
〈표 1〉 멤버쉽 함수

입 력 ($x_2x_1x_0$)	출 력
000	1
001	0
010	1
011	1
100	1
101	0
110	1
111	X

(그림 4)(a)는 <표 1>의 카르노맵을 나타낸다. 카르노맵의 사각형은 최소항이다. 최소화하는 과정은 (그림 4)(b)와 같이 인접해 있는 1이나 X의 최소항들의 블록을 가능한 크게 만든다. 블록이 클수록 항을 구성하는 입력변수의 수가 적어진다. 항을 구성하는 입력변수의 수가 가장 적은 항을 주항(prime term)이라 하며, 주항을 찾는 것이 함수의 최소화이다. <표 1>은 $x_1 + \bar{x}_0$로 최소화된다.



(a) 카르노맵



(b) 주항의 선택

(그림 4) 멤버쉽 함수의 카르노맵과 최소화

(그림 4)에서 부울함수의 최소화와 일괄제거간의 관계를 다음과 같이 해석할 수 있다.

- 0인 최소항은 그룹에서 제거되어야 하는 멤버
- 1인 최소항은 그룹에 남아 있는 멤버
- X인 최소항은 아직까지 할당받지 못한 UID

일괄제거 시에 갱신할 그룹키는 카르노맵에서 가장 적은 수의 주항을 찾는 문제와 밀접한 관계가 있다. 각 주항은 그룹키를 암호화하는데 사용하는 1개의 복합키에 대응하므로 그룹키를 갖는 1개의 메시지에 대응한다. 위의 예에서 그룹키를 갱신하기 위하여, 암호화된 그룹키 $\{GK(r+1)\}_{k_1}$와 $\{GK(r+1)\}_{\bar{k}_0}$를 갖는 2개의 메시지를 그룹에 분배한다. 동시에 제거되는 멤버들의 UID의 공통비트가 많을수록 rekey를 위한 메시지가 적어진다.

Chang모델은 일괄제거시 제거되는 멤버의 UID에 따라 메시지의 수와 복합키를 구성하는 보조키의 수가 달라지며 또한 그룹을 탈퇴하는 멤버들끼리 공모공격을 할 수 있다. 공모공격은 그룹을 탈퇴하는 멤버의 집합이 공모하여 그들이 가지고 있는 키를 조합하면 새로운 키를 불법으로 획득하는 것이다. 예를 들면, 8개의 멤버를 갖는 그룹에서 UID 0과 7이 같은 라운드에 탈퇴한다면 두 멤버는 그룹의 모든 보조키를 보관하고 있기 때문에 공모공격이 가능하다. Chang 모델에서는 공모공격을 막을 수 없으므로 UID를 키의 관리에 사용한 이점을 살리지 못하는 단점이 있다.

2.3 라운드 조정 알고리즘

본 논문에서는 그룹 탈퇴멤버들의 UID를 체크하여 동시에 제거될 멤버들의 라운드를 조정하면 Chang모델에서 발생하는 공모공격문제를 해결할 수 있으며, 그룹키를 암호화하는데 사용하는 복합키의 수를 줄일 수 있으므로 rekey를 위한 메시지의 수를 줄일 수 있다. 또한 복합키를 계산하는 보조키의 수가 적어지게 되어 복합키의 연산비용이 줄어들게 된다.

단계 1 : 제거될 멤버의 UID를 큐에 저장한다. 모든 UID끼리의 해밍거리를 계산한다.
단계 2 : 제거될 멤버들의 공모가능성을 체크한다. 공모가능성이 있으면 공모가능성이 있는 멤버들 간의 해밍거리가 가장 큰 멤버들 중에서 마지막으로 큐에 저장한 멤버를 제거 대상에서 제외한다.
단계 3 : 단계 2의 제거 대상들 간의 해밍거리가 주어진 임계치보다 작은 UID의 멤버만 제거 대상으로 선택한다.
단계 4 : 선택된 UID를 제외한 나머지 그룹멤버들에게 분배할 새로운 그룹키를 암호화할 키를 생성한다.
단계 5 : 선택된 UID는 큐에서 제거된다. 단계 1로 간다.

(알고리즘) 라운드 조정

해밍거리(Hamming distance)[10]은 2개의 UID를 이진 비트로 표현할 때 서로 다른 비트의 수이다. 단계 3에서는 제거될 멤버를 선택하기 위하여 제어기가 임의로 설정한 임계치(threshold)가 적용된다. 제거될 멤버의 해밍거리가 작으면 부울함수에서 더 큰 블록을 만들 수 있으므로 새 그룹키를 암호화하는데 사용되는 복합키의 연산비용을 줄일 수 있으며 rekey를 위한 메시지의 수를 줄일 수 있다. 이것은 동시에 제거되는 멤버들의 UID의 공통비트가 많을수록 그룹의 나머지 멤버들의 공통비트가 많아지므로 rekey를 위한 메시지의 수와 복합키 연산비용이 적어지기 때문이다. 임계치가 작으면 복합키의 연산비용이 적으며 rekey를 위한 메시지의 수가 적어지지만 제거하여할 멤버를 즉시 제거하지 못할 수 있다. 제어기는 연산비용 및 메시지의 수에 대한 효율성과 멤버제거의 즉시성간의 tradeoff에 따라 임

계치를 결정하여야 한다. 메시지의 수에 대한 효율성을 높이기 위해서는 임계치를 높게 설정하며 즉시성을 중요시할 경우에는 임계치를 낮게 설정하여야 한다. 비디오 컨퍼런싱, 비디오서버, 주식정보, 분산게임과 같은 실시간 데이터를 전송하는 응용에서는 임계치를 낮게 설정하고 소프트웨어 분배, 데이터베이스 복사와 같은 비실시간 데이터를 전송하는 응용에서는 임계치를 높게 설정할 수 있다.

UID가 n비트로 표현될 경우, 해밍거리가 n이면 2개의 멤버는 공모가능성이 있다. 2개 이상의 제거될 멤버들의 공모가능성 여부는 다음과 같은 과정에 의해 체크될 수 있다. 제거될 멤버의 UID가 $a_{n-1}a_{n-2}\cdots a_0$, $b_{n-1}b_{n-2}\cdots b_0$, $c_{n-1}c_{n-2}\cdots c_0$,…라 하면 공모 가능성을 일반적으로 식 (4)와 식 (5)와 같이 체크한다.

$$z_i=(a_i+b_i+c_i+\cdots)(a_ib_ic_i\cdots)',\ i=n-1,n-2\cdots 0 \quad (4)$$

$$P=z_{n-1}z_{n-2}\cdots z_1z_0 \quad (5)$$

식 (4)에서 어떤 비트의 위치 i에 대하여, 제거될 멤버들의 UID의 비트 i가 0과 1을 모두 포함하고 있다면 z_i는 1이다. 식 (5)에서 $z_{n-1}z_{n-2}\cdots z_1z_0$는 z_{n-1}에서 z_0까지의 논리곱(AND)을 나타낸다. P가 1이면 공모가능성이 있으며 P가 0이면 공모가능성이 없다.

예를 들어, 3개 멤버의 UID가 각각 $a_3a_2a_1a_0=1100$, $b_3b_2b_1b_0=1010$, $c_3c_2c_1c_0=0101$일 때 UID의 첫 번째 비트들(a_3, b_3, c_3)는 0과 1을 모두 가지고 있으므로 식 (4)에 의해 $z_3=1$이다. 같은 식으로 UID의 두 번째 비트들(a_2, b_2, c_2), 세 번째 비트들(a_1, b_1, c_1)과 네 번째 비트들(a_0, b_0, c_0)도 각각 모두 0과 1을 가지고 있으므로 식 (4)에 의해 $z_2=z_1=z_0=1$이다. 식 (5)에 의해 $P=1$이므로 3개의 멤버는 공모 가능성이 있다.

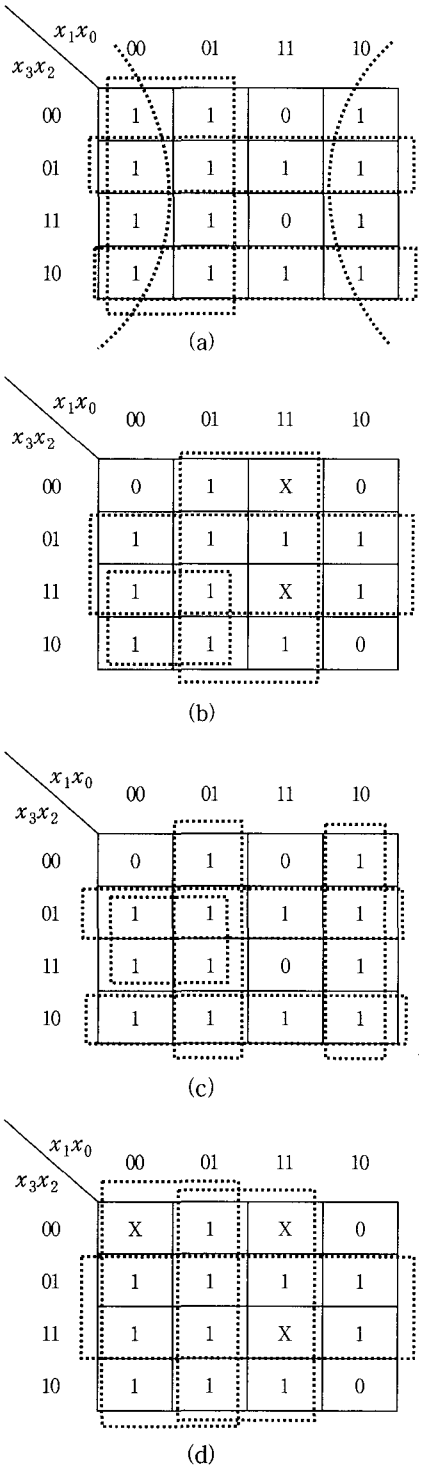
그룹의 크기가 16인 경우, 라운드 조정 알고리즘을 설명하기 위하여 예를 들어 보자. 첫 번째 라운드에서 UID 1111(c_{15}), 0011(c_3), 0000(c_0)을 그룹으로부터 제거하고자 한다. 이들을 큐에 순서대로 입력한다. 큐에 있는 멤버들간의 해밍 거리를 계산하면 <표 2>와 같다.

〈표 2〉 멤버들간의 해밍거리

	1111(c_{15})	0011(c_3)	0000(c_0)
1111(c_{15})	0	2	4
0011(c_3)	2	0	2
0000(c_0)	4	2	0

c_{15}와 c_0은 해밍거리가 4이므로 공모공격가능성이 있다. 2개의 멤버 중에서 c_0이 마지막으로 큐에 입력하였으므로 제거대상에서 제외한다. 제어기가 임계치를 3으로 설정하였다면, c_{15}와 c_3의 해밍거리는 임계치보다 작으므로 c_{15}와 c_3가 함께 제거대상으로 선택되어 첫 번째 라운드에서 그룹으로부터 제거된다. 멤버쉽 함수는 식 (6)과 같다.

$$mem(x_3,x_2,x_1,x_0)=\sum m(0,1,2,4,5,6,7,8,9,10,11,12,13,14) \quad (6)$$



(그림 5) 멤버쉽 함수의 최소화

멤버쉽 함수에 대한 카르노맵은 (그림 5)(a)와 같다. 카르노맵을 최소화하면 $\bar{x}_0+\bar{x}_1+\bar{x}_3x_2+x_3\bar{x}_2$이므로 그룹키를 갱신하기 위하여 4개의 메시지 $\{GK(r+1)\}_{\bar{k}_0}$, $\{GK(r+1)\}_{\bar{k}_1}$, $\{GK(r+1)\}_{f(\bar{k}_3,k_2)}$, $\{GK(r+1)\}_{f(k_3,\bar{k}_2)}$를 분배한다. 여기서 $f(\bar{k}_3,k_2)$는 $\bar{k}_3$와 k_2을 입력으로 하는 함수에 의해 생성된 복합키이다. 일방향 함수 $f(\bar{k}_3,k_2)$는 $\overline{k_3}k_2$와 같이 간단하게 구현되거나 해쉬함수를 사용하여 구현될 수 있다. $\overline{k_3}k_2$는 $GK(r+1)$을 첫 번째 키 $\overline{k_3}$로 암호화한 후 그 결과를 두 번째 키 k_2로 암호화한다는 의미이다. $\{GK(r+1)\}_{\bar{k}_0}$는 $\{c_0,c_2,c_4,c_6,c_8,c_{10},c_{12},c_{14}\}$, $\{GK(r+1)\}_{\bar{k}_1}$는 $\{c_0,c_1,c_4,c_5,c_8,c_9,c_{12},c_{13}\}$, $\{GK(r+1)\}_{f(\bar{k}_3,k_2)}$는 $\{c_4,c_5,c_6,c_7\}$, $\{GK(r+1)\}_{f(k_3,\bar{k}_2)}$는 $\{c_8,c_9,c_{10},c_{11}\}$에 의해 각각 복호화할 수 있다.

두 번째 라운드가 시작하기 전에 새로 c_2, c_{10}이 그룹으로부터 제거되기 위하여 큐에 입력되었다고 가정하자. 현재 큐에는 c_0, c_2, c_{10}이 순서대로 기억되어 있다. 큐에 있는 멤버들간의 해밍 거리를 계산하면 <표 3>과 같다.

〈표 3〉 멤버들 간의 해밍거리

	0000(c_0)	0010(c_2)	1010(c_{10})
0000(c_0)	0	1	2
0010(c_2)	1	0	1
1010(c_{10})	2	1	0

두 번째 라운드에서는 큐에 있는 멤버들 간에 공모가능성이 없고 멤버들 간의 해밍거리가 임계치보다 작으므로 큐에 있는 모든 멤버들이 제거대상으로 선택된다. 멤버쉽 함수는 식 (7)과 같다.

$$mem(x_3,x_2,x_1,x_0)=\sum m(1,4,5,6,7,8,9,11,12,13,14)+\sum d(3,15) \quad (7)$$

멤버쉽 함수에 대한 카르노맵은 (그림 5)(b)와 같다. 카르노맵을 최소화하면 $x_0+x_2+x_3\bar{x}_1$이므로 그룹키를 갱신하기 위하여 3개의 메시지 $\{GK(r+1)\}_{k_0}$, $\{GK(r+1)\}_{k_2}$, $\{GK(r+1)\}_{f(k_3,\bar{k}_1)}$를 분배한다.

Chang모델과 같이 그룹 rekey를 하면 첫 번째 라운드에서는 (그림 5)(c)와 같이 최소화한 결과가 $x_3\bar{x}_2+x_2\bar{x}_1+x_1\bar{x}_0+x_0\bar{x}_1+x_2\bar{x}_3$이므로 rekey를 하기 위하여 5개의 메시지가 필요하다. 또한 가장 큰 문제는 c_0와 c_{15}가 공모공격을 할 수 있다는 것이다. 두 번째 라운드에서의 카르노맵의 최소화는 (그림 5)(d)와 같이 $\bar{x}_0+x_1+x_2$이므로 3개의 메시지가 필요하다.

<표 4>는 위 예에 대하여 본 논문에서 제안한 CAB과 Chang모델간의 rekey를 하기 위하여 뷴배하는 메시지의 수와 복합키를 연산하기 위한 보조키의 수를 비교한 것이다.

〈표 4〉 CAB과 Chang모델간의 메시지의 수와 복합키의 연산비용

	메시지의 수			복합키의 연산비용	
	첫 번째 라운드	두 번째 라운드	합계	1개의 보조키를 갖는 메시지의 수	2개의 보조키를 갖는 메시지의 수
CAB	4	3	7	4	3
Chang	5	3	8	3	5

CAB모델에서는 첫 번째와 두 번째 라운드에서 그룹키를 갱신하기 위하여 총 7개의 메시지를 분배하여야 한다. 그 중에서 1개의 보조키를 사용하여 암호화한 메시지는 4개이며, 2개의 보조키에 의해 생성된 복합키를 사용하여 암호화한 메시지는 3개의 메시지이다.

Chang모델에서는 첫 번째와 두 번째 라운드에서 총 8개의 메시지를 분배하여야 한다. 그 중에서 1개의 보조키를 사용하여 암호화한 메시지는 3개이며, 2개의 보조키에 의해 생성된 복합키를 사용하여 암호화한 메시지는 5개이다.

부울함수에서 주항의 수는 메시지의 수, 주항을 구성하고 있는 리터럴의 수는 그룹키를 암호화하는데 사용할 보조키의 수이다. 주항을 구성하는 리터럴의 수가 적으면 복합키를 생성하는데 필요한 연산의 비용을 줄일 수 있다. CAB모델은 Chang모델보다 메시지의 수와 복합키를 생성하는 연산 비용을 줄일 수 있다.

제거될 멤버들간의 해밍거리가 작을수록 그룹에 남아있는 멤버들을 더 큰 블록으로 묶을 수 있게 되어 블록(주항)의 수가 줄어들거나 주항의 리터럴수가 줄어들게 된다. 이것은 그룹키를 암호화하는 메시지의 수와 복합키의 연산비용이 줄어들게 된다는 의미와 같다.

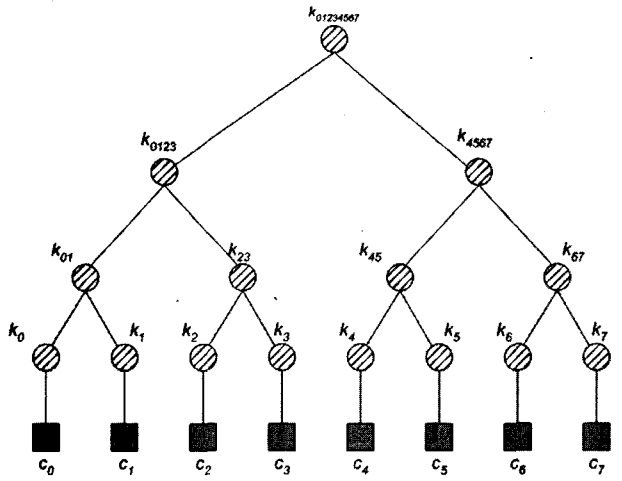
3. 비교분석

본 논문에서 제안한 모델(CAB, Collusion Attack Block)과 기존의 좋은 확장성을 보여준 모델인 Wong, Chang의 모델을 비교분석함으로서 CAB모델의 성능이 우수함을 보인다. 3개의 모델 간에 최악의 메시지의 수, 제어기의 키의 수, 멤버의 키의 수, 복합키의 연산비용, 공모여부에 대하여 비교한다.

최악의 메시지의 수는 멤버 탈퇴시 rekey를 위하여 최악의 경우에 분배하여야 하는 메시지의 수를 나타내며 확장성의 중요한 척도이다. 제어기는 그룹키 관리를 위하여 모

든 멤버들이 가지고 있는 키들을 생성하여 분배하여야 한다. 제어기의 키의 수는 rekey를 위하여 제어기가 관리하여야 하는 키의 수를 나타낸다. 멤버의 키의 수는 그룹키를 관리하기 위하여 각 멤버가 제어기로부터 받아서 관리하여야 하는 키의 수를 나타낸다. 복합키의 연산비용은 그룹키를 암호화하는데 필요한 복합키를 연산하는 비용을 나타내며, 공모여부는 탈퇴하는 멤버들이 공모하여 그룹키를 불법으로 가로챌 가능성이 있는지를 나타낸다.

(그림 7)은 그룹의 크기가 8인 Wong모델의 키 관리체제를 나타낸다. 트리에서 빗금친 원형 노드는 개인키 및 보조키를, 사각형 노드는 멤버를 나타내며 트리의 루트노드는 그룹키를 나타낸다. Wong모델에서는 멤버가 가입을 하면 안전한 채널을 사용하여 멤버에게 유일한 개인키(private key)를 분배한다. 또한 키트리의 단말노드에서 루트노드까지 있는 보조키와 그룹키를 분배한다. 안전한 채널은 Diffie-Hellman[11] 키교환 방식을 사용할 수 있다. Wong모델은 CAB와 Chang모델과 달리 개인키를 분배하여야 하므로 그룹의 크기가 클 때 그룹제어기가 관리하여야 하는 키의 수가 많아진다.



(그림 7) Wong모델의 키 관리체제

1개의 멤버가 제거될 때 트리의 단말노드로부터 루트노드까지의 모든 키가 변경되어야 한다. 갱신된 키는 그룹의 각 멤버들에게 유니캐스트하는 것이 아니라 그룹에 분배된다. 예를 들면, 멤버 c_5가 그룹을 떠날 때, k_{45}, k_{4567}, $k_{01234567}$이 갱신되어야 한다. k_{45}는 k_4에 의해 암호화되며, k_{4567}은 k_{67}과 새로 갱신된 k_{45}에 의해 암호화된다. 또한 $k_{01234567}$은 k_{0123}과 새로 갱신된 k_{4567}에 의해 암호화되어 전체 그룹에 분배된다.

크기가 N인 Wong모델에서 1개의 멤버를 제거할 때, rekey를 위하여 $2\log N-1$개의 메시지를 분배하여야 하며, 각 멤버는 $\log N+1$개의 키를 관리하여야 한다. 그리고 제어기는 $2N-1$개의 키를 가지는 트리를 관리하여야 한다. 그룹의 크기가 8일 때 Wong모델은 15개의 키(8개의 개인키, 6개의 보조키, 1개의 그룹키)를 관리하며, Chang과 CAB모델은 7개의 키(6개의 보조키, 1개의 그룹키)를 관리한다.

멤버의 키의 수는 Wong, Chang과 CAB모델이 각각 $\log N+1$, $\log N$과 $\log N$이다. 제어기가 관리하는 키의 수는 Wong, Chang과 CAB모델이 각각 $2N-1$, $2\log N+1$와 $2\log N+1$이다. 1개의 멤버를 제거할 때 분배하는 메시지의 수는 Wong, Chang과 CAB모델의 경우 각각 $2\log N-1$, $\log N$과 $\log N$이다.

여러 개의 멤버를 제거할 때 최악의 메시지의 수를 비교하여 보자. Wong모델에서 최악의 메시지의 수는 키 트리의 최하의 모든 서브 트리에 있는 2개의 멤버중 1개가 탈퇴할 경우이다. 이때 키 트리의 모든 보조키가 갱신되어야 하므로 최악의 메시지의 수는 $N-2$이다. Chang모델에서 최악의 메시지의 수는 각 멤버간의 해밍거리가 2이상인 $N/2$개의 멤버가 동시에 제거될 때이며 최악의 메시지의 수는 $N/2$이다[4]. CAB모델에서 최악의 메시지의 수는 라운드 조정 알고리즘을 수행하므로 Chang모델의 최악의 메시지의 수보다 적다.

<표 5>는 그룹에서 1개의 멤버가 제거될 때, rekey를 하기 위하여 제어기가 분배하는 최악의 메시지의 수, 제어기가 관리하는 키의 수, 멤버가 관리하는 키의 수에 대하여 Wong, Chang, CAB모델간의 비교를 보여 주고 있다. 또한 <표 6>은 여러 개의 멤버가 제거될 때, 제어기가 분배하는 최악의 메시지의 수, 제어기가 관리하는 키의 수, 멤버가 관리하는 키의 수, 복합키를 생성하기 위한 연산비용, 공모공격 가능성에 대하여 3개의 모델간의 비교를 보여주고 있다.

〈표 5〉 한 개의 멤버 제거시 모델간의 비교

	최악의 메시지의 수	제어기의 키의 수	멤버의 키의 수
Wong	$2\log N-1$	$2N-1$	$\log N+1$
Chang	$\log N$	$2\log N+1$	$\log N$
CAB	$\log N$	$2\log N+1$	$\log N$

〈표 6〉 여러 개의 멤버 제거시 모델간의 비교

	최악의 메시지의 수	제어기의 키의 수	멤버의 키의 수	복합키 연산비용	공모공격 가능성
Wong	$N-2$	$2N-1$	$\log N+1$	N/A	NO
Chang	$N/2$	$2\log N+1$	$\log N$	> CAB	YES
CAB	< $N/2$	$2\log N+1$	$\log N$	< Chang	NO

제거될 멤버들간의 해밍거리가 클수록 많은 메시지의 수와 메시지를 암호화하는 보조키의 수가 많이 요구된다. CAB 모델은 라운드 조정알고리즘에 의해 제거될 멤버들간의 해밍거리가 임계치 이하인 멤버만 제거하므로 Chang모델의 최악의 메시지의 수보다 적으며 메시지를 암호화하는데 필요한 보조키의 수가 적으므로 Chang모델보다 키 연산 비용이 적게 소요된다. 또한 Chang모델에서 발생하는 공모공격의 문제점을 해결할 수 있다. <표 6>에서 복합키를 생성하기 위한 연산비용 항목은 CAB와 Chang모델간의 비교만 가능하나 Wong모델과의 비교는 불가능하다.

새로운 키를 생성하는 것은 많은 연산을 요구하므로 re-key를 하기 위하여 보내는 메시지의 수를 줄이거나 복합키를 생성하기 위한 연산을 줄이는 것은 키 관리체제를 평가하는 중요한 척도이다. 본 논문에서 제안한 모델은 여러 개의 멤버들을 제거할 때 라운드 조정 알고리즘을 수행하기 때문에 rekey를 하기 위한 메시지의 수와 복합키의 연산의 비용을 줄이며 공모공격의 문제점을 해결할 수 있는 이점이 있다.

4. 결 론

제어기는 rekey를 위하여 그룹 멤버에게 그룹키를 보낸다. 그룹키는 그룹의 모든 멤버들이 공유하며, 멀티캐스트 그룹에게 메시지를 보낼 때 암호화하거나 복호화할 때 사용된다. 또한 멤버는 보조키 세트들 중에서 1개의 세트를 받는다. 보조키는 그룹키를 안전한 방법으로 갱신하는데 사용된다.

Rekey할 때 제거될 멤버들간의 해밍거리가 클수록 많은 메시지와 메시지를 암호화하는 보조키의 수가 많이 요구된다. CAB모델은 라운드 조정알고리즘에 의해 제거될 멤버들간의 해밍거리가 임계치 이하인 멤버만 제거하므로 Chang 모델의 최악의 메시지의 수보다 적게 가질 수 있으며 메시지를 암호화하는데 필요한 보조키의 수가 적으므로 Chang 모델보다 복합키 연산 비용이 적게 소요된다. 또한 Chang 모델에서 발생하는 공모공격의 문제점을 해결할 수 있었다.

기존의 좋은 확장성을 보여준 모델인 Wong, Chang모델과 CAB모델을 비교분석함으로서 CAB모델의 성능이 우수함을 보였다. 3개의 모델간에 최악의 메시지의 수, 제어기의 키의 수, 멤버의 키의 수, 복합키의 연산비용, 공모여부에 대하여 비교하였다.

CAB모델의 멤버의 키의 수는 CAB와 Chang모델이 Wong보다 적으며, 제어기가 관리하는 키의 수는 Wong모델보다 적었다. 또한 CAB모델의 경우 1개의 멤버를 제거할 때 분배하는 메시지의 수는 Wong모델보다 적었으며, 여러 개의 멤버를 제거할 때 최악의 메시지의 수는 Wong과 Chang모델보다 적었다.

새로운 키를 생성하는 것은 많은 연산을 요구하므로 re-key를 하기 위하여 보내는 메시지의 수를 줄이거나 복합키를 생성하기 위한 연산을 줄이는 것은 키 관리체제를 평가하는 중요한 척도이다. 본 논문에서 제안한 모델은 여러 개의 멤버들을 제거할 때 라운드 조정 알고리즘을 수행하기 때문에 rekey를 하기 위한 메시지의 수와 복합키의 연산의 비용을 줄이며 공모공격의 문제점을 해결할 수 있는 이점이 있었다.

임계치에 따라 rekey의 비용을 산출하는 실험연구가 필요하며, 또한 그룹멤버인 공격자와 이미 그룹을 떠난 공격자들이 서로 공모하여 그룹멤버인 공격자가 현재 알고 있는 그룹키를 그룹을 떠난 공격자에게 제공하는 공모 공격에 대한 차단기법을 향후에 연구할 필요가 있다.

참 고 문 헌

[1] T. Hardjono, G. Tsudik, "IP Multicast Security : Issues and Directions," Annales De Telecom, pp.324-340, July-August, 2000.

[2] A. Ballardie, "Scalable Multicast Key Distribution," RFC 1949, May, 1996.

[3] D. Wallner, E. Harder, R. Agee, "Key Management for Multicast : Issues and Architectures," RFC 2627, 1999.

[4] I. Chang, R. Engel, D. Kandlur, D. Pendarakis, D. Saha, "Key Management for Secure Internet Multicast using Boolean Function Minimization Techniques," Proceedings of Infocom, New York, March, 1999.

[5] C. K. Wong, M. Gouda, S. S. Lam, "Secure Group Communications using Key Graphs," Proceedings of ACM SIGCOMM, Vancouver, British Columbia, Sep., 1998.

[6] S. Banerjee, B. Bhattacharjee, "Scalable secure Group Communication over IP Multicast," Proceedings of ICNP, Nov., 2001.

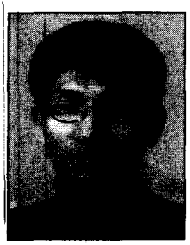
[7] M. Steiner, G. Tsudik, M. Waidner, "Key Agreement in Dynamic Peer Groups," IEEE Tr. on Parallel and Distributed Systems, Vol.11, No.8, pp.769-780, 2000.

[8] X. S. Li, Y. R. Yang, M. G. Gouda, S. S. Lam, "Batch Rekeuing for Secure Group Communications," WWW10, May, 2001.

[9] J. F. Wakerly, "Digital Design Principles and Practices," Prentice-Hall, 1990.

[10] http://www.nist.gov/dads/HTML/hammingdist.html.

[11] W. Diffie and M. Hellman, "New Directions in Cryptography," IEEE Tr. on Information Theory, IT-22, pp.644-654, Nov., 1976.



김 태 균

e-mail : skyeedu@empal.com

1989년 충남대학교 기술교육학과(기술전공) (공학사)

2003년 한국교원대학교 컴퓨터교육과 (교육학석사)

1989년~1990년 병천고등학교 교사

1990년~1995년 대천고등학교 교사

1995년~2000년 천안중앙고등학교 교사

2000년~2001년 예산전자공업고등학교 교사

2001년~2003년 한국교원대학교 파견

2003년~현재 병천고등학교 교사(공주대학교 박사과정)

관심분야 : 컴퓨터교육, 정보보안, 네트워크

정 종 인

e-mail : jichung@kongju.ac.kr

1981년 경북대학교 전자공학과(전산전공) (공학사)

1985년 경북대학교 대학원 전자공학과 (공학석사)

1995년 서강대학교 전자계산학과(공학박사)

1985년~1997년 우송공업대학 전산과 교수

1999년~2000년 미국 USC post-doc.

1997년~현재 공주대학교 컴퓨터교육과 교수

1999년~현재 멀티미디어기술사

관심분야 : 정보보안, 병렬처리구조, 네트워크