이학석사 학위논문

안전한 멀티캐스트를 위한 효율적인 그룹키 관리구조

지도교수 이 경 현

이 논문을 이학석사 학위논문으로 제출함

2002년 2월

부경대학교 대학원

전자계산학과

박 영 호

박영호의 이학석사 학위논문을 인준함

2001년 12월 26일

주 심 공학박사 김 창 수 

위 원 공학박사 김 영 봉 

위 원 이학박사 윤 성 대 

<차 례>

<표차례> ii
<그림차례> iii
Abstract iv
1. 서론 1
2. 멀티캐스트 보안 개요 3
2.1 멀티캐스트 보안요소 3
2.2 멀티캐스트 보안 기법 설계를 위한 고려사항 4
2.3 안전한 멀티캐스트 보안 구조 6
2.4 그룹키 관리를 위한 고려사항 7
3. 안전한 멀티캐스트를 위한 그룹키 관리 기법 10
3.1 단순 그룹키 공유기법 11
3.2 트리 기반의 그룹키 관리 기법 12
3.3 GDH(Group Diffie-Hellman)를 이용한 그룹키 관리 21
4. 확장성을 위한 그룹 관리키 구조 방안 28
4.1 2-계층(2-tier) 방식의 그룹 관리 구조 28
4.2 그룹 운용(Group operation) 방식 30
4.3 그룹 합병 및 분할 34
4.4 구현방안 및 고찰 38
4.5 활용방안 43
5. 결론 48
참 고 문 헌 49

<표 차 례>

〔표 1〕 멀티캐스트의 보안 영역 4
〔표 2〕 어플리케이션 유형에 따른 보안 고려사항 6
〔표 3〕 OFT 키 갱신에 대한 계산량 비교 21
〔표 4〕 NAGKA와 AGKA 계산량 25
〔표 5〕 키 설정 시간 비교 40
〔표 6〕 키 갱신 계산량 44

<그림 차례>

〔그림 1〕 멀티캐스트 보안 시스템 구조 7
〔그림 2〕 단순 그룹키 공유 방식 11
〔그림 3〕 n=8, k=3인 키 트리 13
〔그림 4〕 u_9의 그룹 참여시 키 트리 13
〔그림 5〕 OFT의 노드 키 계산 17
〔그림 6〕 OFT 트리의 구성 17
〔그림 7〕 OFT의 멤버 참여/탈퇴에 대한 키 트리 갱신 18
〔그림 8〕 NAGKA의 그룹키 계산 22
〔그림 9〕 TGDH의 멤버 참여에 대한 키 트리 갱신 26
〔그림 10〕 TGDH의 멤버 탈퇴대한 키 트리 갱신 27
〔그림 11〕 2-계층 형태의 그룹 관리 구조 28
〔그림 12〕 SGC의 상부계층 탈퇴 33
〔그림 13〕 SG_i와 SG_j의 키 트리 구성 34
〔그림 14〕 그룹 합병으로 인한 키 트리 갱신 35
〔그림 15〕 서브그룹 SG_i의 그룹 분할 이전 37
〔그림 16〕 서브그룹 SG_i의 그룹 분할 38
〔그림 17〕 그룹 관리 시스템 구조 39
〔그림 18〕 OFT 키 트리 계산 시간 41
〔그림 19〕 TGDH 키 트리 계산 시간 41
〔그림 20〕 멤버 추가시 TGDH 갱신 시간 42
〔그림 21〕 멤버 삭제시 TGDH 갱신 시간 42
〔그림 22〕 이동 네트워크에서의 그룹 구조 44
〔그림 23〕 Hand-over 과정에서의 키 설정 45

Scalable Group Key Management Structure for Secure Multicast

Young-Ho Park

Dept. of Computer Science, Graduate School,
Pukyong National University

Abstract

The development of the Internet has led to emerging the new Internet applications based on a group communication model and multicast is suitable protocol for group communication but does not provide security primitives. Thus, group communications must satisfy the security primitives like a confidentiality, integrity and authenticity for secure group communication. Compared to two-party communications, group communication has a unique characteristic, group dynamics, that group membership can change over time. Therefore, forward secrecy for newly joining member and backward secrecy for left member must be considered. In addition, an efficient group key management is required for a membership change.

In this paper, we propose a scalable group key management structure for secure multicast and discuss its implementation issues. The whole group is layered into 2-tier composed with control group as 1st-tier and data group as 2nd-tier. Data group is the set of subgroups where members are resided in and managed by each subgroup manager while control group is consisted with group manager and all subgroup manager. The task for group management is shared among the subgroup managers, and a membership change in one subgroup does not affect another subgroups. Moreover, group merge in our research makes it possible to recovery from system fault of subgroup manager by migrating members to well subgroup.

1. 서론

인터넷의 활용 영역이 다양한 분야로 확대되면서 안전한 데이터의 전송이 많은 어플리케이션의 주요 요구사항으로 대두되었다. 그리고 여러 명의 사용자들에게 동일한 서비스를 제공해주기 위한 그룹 통신에 대한 요구도 증가하는 추세이다. 그룹 통신에는 뉴스나 주식 정보 제공 서비스나 원격 회의(tele-conference), 인터넷 방송, 소프트웨어 업데이트와 같은 서비스 형태를 가진다. 이러한 그룹 통신은 일대다(one-to-many) 혹은 다대다(many-to-many) 통신 형태로 구성되며, 멀티캐스트(multicast)는 이러한 그룹 통신에 적합한 프로토콜이다.

멀티캐스트는 송신측의 전송 오버헤드를 줄이고 네트워크 대역폭을 감소시키면서 그룹통신에 효율적인 기능을 제공해주지만 보안에 관련된 사항을 고려하지 않고 있으므로, 안전한 그룹 통신을 위해 그룹 사용자들에게 전송되는 정보에 대한 기밀성(confidentiality)과 무결성(integrity) 그리고 그룹 사용자에 대한 인증과 같은 보안 요구사항이 만족되어야 한다. 이러한 보안 요구사항을 만족하기 위해 기존의 IPSec이나 SSL(Secure Socket Layer)같은 일대일(one-to-one) 형태의 안전한 통신을 위한 보안 메커니즘을 적용하여 구현할 수 있으나, 이러한 보안 메커니즘을 그룹 통신에 동일하게 적용할 경우 각 사용자마다 별도의 보안 기능을 적용함으로써 n명의 사용자들에 대해 n개의 세션을 관리해야하므로 $O(n)$의 복잡성을 야기하게 된다.

그러므로 안전한 그룹 통신 프로토콜은 그룹 멤버십의 제어와 효율적이며 안전한 그룹키의 관리를 위한 기능을 제공해야 한다. 또한 일대일 통신이 하나의 통신 세션동안 정적인데 반해 그룹 통신은 한 세션동안 사용자들의 그룹 참여와 탈퇴가 발생하는 동적인 특성을 고려해야 한다. 즉, 새로 그룹에 참여한 사용자는 이전에 전송되었던 그룹 데이터에 접근하지 못 하는 backward secrecy를 제공해야 하고, 기존의 사용자가 그룹을 탈퇴하는 경우 이후의 그룹 데이터에 접근하지 못 하는

forward secrecy를 제공해야 한다. 그리고 그룹키의 갱신과 데이터의 전송 및 암호화와 관련된 오버헤드는 멀티캐스트 그룹의 크기에 독립적이어야 하며, 멀티캐스트 그룹에서의 호스트의 추가와 삭제로 인한 영향이 그룹의 모든 멤버에게 끼치지 않도록 해야한다.

본 논문에서는 그룹키 관리에 있어서 확장성(Scalability)을 제공하기 위한 그룹키 관리구조를 제안하고 이에 대한 설계방안과 활용방안에 대해서 논의하고자 한다. 그룹키 관리의 효율성과 확장성을 위해 전체 그룹구조를 실제 멤버들의 서브그룹(subgroup)으로 구성되는 하부계층(2nd-tier)과 서브그룹의 관리자들을 제어하기 위한 상부의 관리계층(1st-tier)으로 구성되는 2-계층(2-tier)형태의 구조를 구성한다. 그리고 그룹 관리자의 그룹키 관리에 대한 부담(workload)을 각 서브그룹 관리자들에게 분담하고, 각 서브그룹은 자신들만의 자치영역을 구성함으로써 해당 그룹의 멤버십 변경에 대한 영향을 지역적으로 한정시킴으로써 다른 서브그룹에 영향을 주지 않도록 한다.

그리고 각 서브그룹간의 그룹 병합(merge)과 분할(partition)을 통해 서브그룹 관리서버에 시스템 오류가 발생하거나 작업량이 과다한 경우 다른 서브그룹 관리서버로 멤버들을 이전시킴으로써 계속적인 멀티캐스트 서비스를 제공하고 작업량을 분담할 수 있다.

2. 멀티캐스트 보안 개요

멀티캐스트는 멀티캐스트 그룹에 참여한 여러 명의 수신자들에게 동일한 데이터를 전송하기 위한 서비스이다. 멀티캐스트는 Class D로 예약된 224.0.0.0부터 239.255.255.255까지의 IP 주소를 사용하며, 송신측의 전송 오버헤드와 네트워크 대역폭을 감소시키는 이점을 제공한다. 멀티캐스트가 효율적인 그룹통신 메커니즘을 제공하지만 보안에 대한 문제는 고려하지 않고 있다. 즉, 멀티캐스트 데이터는 여러 명의 수신자들에게 전송되기 위해 다중의 전송 채널을 통해 전송되므로 악의적인 도청의 기회를 더 많이 제공하게 되고, 멀티캐스트 주소는 공개적이므로 누구나 멀티캐스트 그룹으로 전송되는 데이터에 접근할 수 있다. 또한 부당한 사용자가 멀티캐스트 그룹으로 부적절한 데이터를 전송할 경우 네트워크 트래픽의 증가로 인한 혼잡을 야기할 수 있으므로 정당한 사용자들간의 그룹 통신을 위한 보안 기능이 제공되어야 한다.

2.1 멀티캐스트 보안 요소(Security Issues in Multicast)

안전한 그룹 통신을 위한 멀티캐스트의 보안 영역은 [표 2-1]과 같이 핵심 보안 영역(Core Problem Area)과 기반구조상의 영역(Infrastructure Problem Area) 그리고 어플리케이션 영역(Complex Application Area)으로 구분할 수 있다[1].

이러한 보안 문제들은 암호학적인 메커니즘들을 적용함으로써 해결할 수 있으며, 인증(authentication), 데이터 기밀성(confidentiality)과 무결성(integrity)을 제공해야 한다. 기밀성은 안전한 멀티캐스트 통신을 위한 가장 기본적인 사항으로, 그룹 멤버들간에 암호화에 사용되는 키를 공유함으로써 제공될 수 있다. 그러므로 그룹키의 관리가 안전한 멀티캐스트를 위한 핵심 사항이라 할 수 있으며, 허가된 사용자들만이 그룹키에 접근하도록 하기 위해 인증 기능을 제공해야 한다. 인증 기능은

정당한 사용자의 그룹 참여를 허용하고, 안전한 멀티캐스트 통신을 위한 데이터의 접근을 허가된 사용자들로 제한하도록 한다. 그리고 데이터가 전송도중 변조되는 것을 막기 위해 데이터에 대한 무결성도 제공해야 한다. 표 1은 멀티캐스트의 보안 영역을 나타내고 있다.

[표 1] 멀티캐스트 보안 영역

Core Problem Area	● 멀티캐스트 데이터의 기밀성, 무결성, 인증 ● 멀티캐스트 그룹 키 관리 ● 그룹 보안과 관련된 정책의 설정
Infrastructure Problem Area	● 멀티캐스트 라우팅 프로토콜상의 보안 ● 멀티캐스트 프로토콜의 신뢰성
Application Problem Area	● 그룹 키 생성과 분배(Key Agreement) ● 멤버들간의 안전한 통신 ● 그룹과 그룹 멤버의 인증(Authentication) ● 멤버의 부인봉쇄(Non-repudiation) ● 멤버의 익명성 ● 공격에 대한 강건성(Robustness)

2.2 멀티캐스트 보안 기법 설계를 위한 고려사항

멀티캐스트 보안의 목적은 모든 그룹 통신에 있어서 인증과 비밀성을 유지함으로써 그룹에 가입한 허가된 사용자만이 데이터를 전송하고 수신하도록 하는 것이다. 멀티캐스트를 통한 안전한 그룹 통신을 제공하기 위해 유니캐스트(unicast) 통신에서 사용되는 보안 메커니즘을 그대로 멀티캐스트에 적용할 수 있으나, 유니캐

스트 보안 메커니즘을 그대로 적용할 경우 n개의 세션을 관리해야 하므로 통신상의 비용과 계산상의 비용이 $O(n)$으로 증가하게 되어 효율성이 저하된다. 또한 멀티캐스트의 경우 하나의 통신 세션동안 그룹 멤버의 가입과 탈퇴가 빈번히 발생하는 동적인 성질을 가지므로, 멀티캐스트 보안 메커니즘을 설계함에 있어서 그룹의 동적인 특성과 시스템의 효율성 및 확장성을 함께 고려해야 한다. 그룹의 확장성은 그룹의 동적 성질과 함께 고려되어야 하는 사항으로, 그룹 멤버의 수가 증가하더라도 그룹 관리자의 작업 부담은 동일해야 함을 의미한다. 즉, 그룹의 크기가 커지더라도 데이터의 전송과 암호화를 위한 그룹의 관리가 효율적이어야 한다.

안전한 그룹 통신을 위한 멀티캐스트 보안 기법은 데이터의 암호화를 위해 그룹 멤버들간에 암호화키를 분배하고 유지하기 위한 그룹키 관리 솔루션을 필요로 하고 더불어 수신된 데이터가 정당한 송신자로부터 전송된 것인지를 검증하기 위한 인증기법도 필요로 한다. 안전한 멀티캐스트를 위한 보안 기법은 멀티캐스트 어플리케이션의 형태와 그룹의 특성에 따라 보안 요구사항의 차이가 있을 수가 있기 때문에, 멀티캐스트의 보안에서는 데이터 기밀성과 인증을 분리해서 다루기도 한다.

어플리케이션 형태는 멀티캐스트 그룹 통신의 유형에 따라 일대다(one-to-many) 또는 다대다(many-to-many)로 구분할 수 있다. 일대다 통신은 뉴스나 주식정보 또는 PPV(Pay Per View)처럼 단일 송신측과 다중의 수신측으로 구성되는 그룹 통신 형태이다. 뉴스나 주식정보와 같이 공개적으로 알려질 수 있는 데이터에 대해서는 데이터의 기밀성보다는 데이터를 보낸 송신자에 대한 인증이 중요시되어야 하고, 인터넷 방송과 같은 PPV형태의 어플리케이션의 경우 서비스 사용료를 지불하는 정당한 가입자만이 데이터에 접근할 수 있어야 하므로 데이터의 기밀성과 송신자의 인증이 모두 고려되어야 한다.

다대다 통신에는 다자간 채팅(multi-party chatting)이나 다자간 회의(multi-party conference) 같은 어플리케이션이 있다. 회의에 있어서 모든 통신은 현재 세션에 참여한 멤버들간에만 이루어져야 하므로 데이터에 대한 기밀성과 송신자 인증이 함께 고려되어야 하며, 채팅처럼 사용자의 익명성이 요구되는 어플리케이션

에는 인증을 고려하지 않을 수도 있다.

[표 2] 어플리케이션 유형에 따른 보안 고려사항

유형 / 기능	일대다(one-to-many)		다대다(many-to-many)	
	PPV	뉴스, 주식시세	원격지 회의	채팅
인증	○	○	○	△
기밀성	○	△	○	○
무결성	○	○	○	○

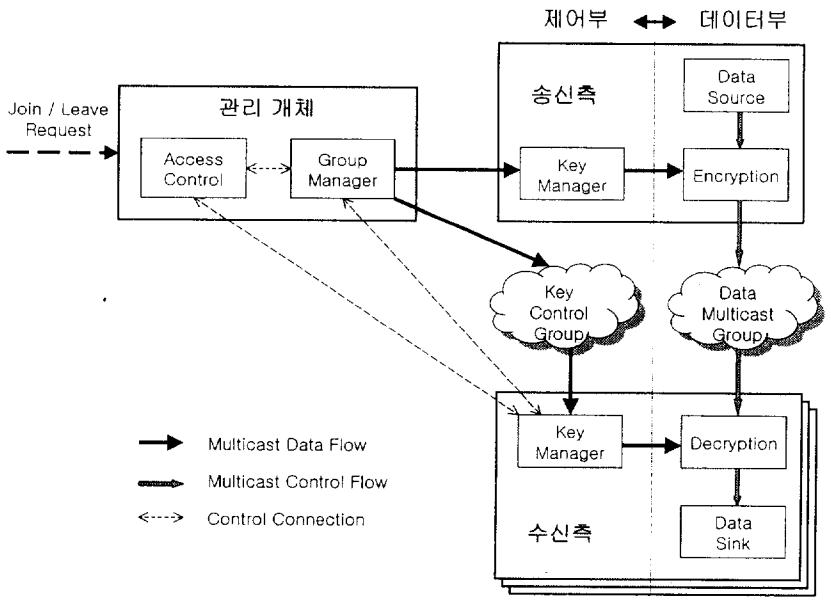
대부분의 안전한 그룹 통신에 관한 연구들이 안전한 그룹의 구조(secure group architecture)[2][3]과 그룹 키 관리(group key management)[4][5][6] 문제에 초점을 두고 수행되어 왔으며 최근 멀티캐스트 패킷의 인증(Packet authentication)[7]에 대한 연구도 주요 대상이 되고 있다.

2.3 안전한 멀티캐스트 보안 구조

안전한 멀티캐스트를 위한 보안시스템의 구조는 그림 1과 같다. 그룹 관리자는 자신의 그룹에 대한 보안 정책(Security Policy)을 설정하고, 사용자들의 그룹 참여와 탈퇴를 그룹의 보안 정책에 따라 결정하게 된다. 그룹의 보안 정책은 또한 키의 생성과 분배 그리고 키의 갱신에 대한 정책도 설정하며, 키 관리자(key manager)가 그룹의 정책에 따라 키 관리를 수행하게 된다.

그림 1에서 시스템은 실질적인 데이터에 관여하는 데이터부(data plane)와 키 관리에 관여하는 제어부(control plane)로 구성된다. 제어부의 키 관리 프로세스는 안전한 그룹 통신을 위한 키 생성(key generation)과 키 협정(key agreement) 그리

고 키 분배(key distribution)를 포함한다.



[그림 1] 멀티캐스트 보안 시스템 구조

송신자가 그룹으로 안전한 데이터 전송을 위해 그룹키를 통해 전송하고자 하는 데이터를 암호화 한 후 그룹으로 전송하게 된다. 수신자는 해당 멀티캐스트 채널을 통해 수신된 데이터를 그룹키로 복호화 함으로써 데이터에 접근할 수 있다. 그러므로 안전한 그룹 통신을 위해서는 그룹키의 관리가 핵심이라 할 수 있다.

2.4 그룹 키 관리를 위한 고려사항

멀티캐스트 보안의 핵심은 그룹키의 관리이며, 그룹키 관리기법의 설계시 고려할 사항으로 시스템의 보안성(secrecy)과 관리를 위한 시스템의 성능 측면을 모두 고려해야 한다. 그룹키 관리기법의 설계를 위한 보안측면의 고려사항은 다음과 같

다.

① 그룹키의 비밀성(Group Key Secrecy)

가장 기본적인 성질로서, 어떤 악의적인 공격자가 그룹키를 도출하는 것이 계산상 불가능하여야 한다.

② Forward Secrecy

악의적인 공격자가 이전 세션의 그룹키들에 대한 정보를 알고있더라도 이후의 그룹키를 계산하지 못 함으로써 데이터에 접근할 수 없어야 한다.

③ Backward Secrecy

악의적인 공격자가 이후에 알려진 그룹키에 대한 정보를 가지고서 이전 세션의 그룹키를 계산하지 못 함으로써 데이터에 접근할 수 없어야 한다.

④ 키 독립성(Key Independency)

그룹키 집합 K의 적당한 부분집합 K'을 알고있는 공격자가 다른 그룹키의 부분집합 $\overline{K} \in (K - K')$를 계산할 수 없어야 한다.

위 네 가지 성질들은 서로 연관성을 가지고 고려되어야 한다. 그룹 멤버의 탈퇴시 이후의 데이터에 대한 forward secrecy를 제공해야 하며 새로운 멤버의 참여서 이전의 데이터에 대한 backward secrecy를 제공해야 한다. 그리고 여러 멤버들의 공모로 인해 그룹키가 노출되는 것을 막기 위해 키의 독립성도 제공해야 한다.

이러한 암호학적인 성질들과 함께 그룹키 관리 메커니즘을 구현함에 있어 시스템의 성능에 대해서도 다음과 같은 사항을 고려해야 한다.

① 시스템 구조 : one point failure의 해결

즉 한 개체의 오류가 전체 시스템의 오류로 확산되면 안 된다.

② 시스템의 확장성(Scalability)

다수의 멤버들이 광범위한 지역으로 분포되어 있는 그룹을 관리할 수 있도록 확장성을 제공해야하고, 동적인 멤버십을 가지는 그룹의 빈번한 키 갱신을 처리할 수 있어야 한다.

③ 그룹 관리자와 멤버들이 관리하는 키의 개수

④ 키 갱신을 위해 필요한 메시지의 개수

⑤ 키 관리를 위한 processing time

3. 안전한 멀티캐스트를 위한 그룹키 관리 기법

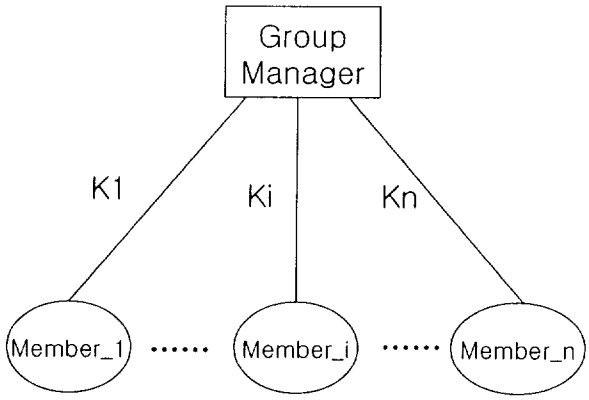
멀티캐스트 그룹 멤버들이 광범위한 지역으로 분포되어 있는 환경에서 메시지를 안전하게 전달하고 그룹을 효율적으로 관리하기 위한 연구들의 대부분이 그룹키의 관리에 초점을 맞추어 연구 되어왔다. 이들은 시스템 관리 형태에 따라 크게 중앙집중형 방식과 분산형 방식 그리고 계층적 방식으로 구분할 수 있다.

중앙집중형 방식은 어떤 하나의 신뢰되는 개체(trusted entity)가 모든 그룹의 멤버들에게 암호화키의 분배를 담당하는 방식으로, 그룹의 멤버십이 변경될 때마다 그룹 관리자가 갱신된 그룹키를 모든 그룹 멤버들에게 안전하게 전달하는 방식이다.

분산형 방식은 그룹의 모든 멤버들이 동등하게 신뢰되는 방식으로 그룹키 분배에 대한 책임을 여러 멤버들에게 분담하는 방식이다. 그러나 키의 분배를 멤버들에게 분담함으로 인해 내부 공모에 대한 보안상의 취약성을 가질 수 있다. 계층적 방식은 시스템의 확장성을 제공하기 위해 키 분배 트리라는 논리적 구조를 사용하여 암호화키를 분배하는 방식으로, 이 방식은 다시 두 가지로 구분된다. 첫 번째 방식은 키들간의 계층구조를 사용하는 방식이고 두 번째는 확장성을 위해 네트워크 노드들의 계층구조를 사용하는 방식이다[2][8][10]. Iolus[2]는 그룹 키 관리에 있어서 확장성을 제공하기 위한 안전한 멀티캐스트 그룹 구조에 대한 연구로, 멀티캐스트 그룹을 네트워크 상의 노드들간의 계층구조로 분할함으로써 키관리에 효율성을 제공하도록 하였다. 키 트리를 사용하는 대부분의 방식이 중앙집중형의 키 서버의 확장성을 위해 제안되었으며[4][5][6], [9]와 [12]에서는 키 트리를 이용한 분산형 키 관리 방안을 제안하였다.

3.1 단순 그룹키 공유 기법

이 방법은 중앙집중식의 키 관리방식으로, 그림 2와 같이 그룹 관리서버가 n명의 멤버에 대해 각각의 멤버와 비밀키를 공유하는 방식이다. 어떤 멤버의 참여와 탈퇴에 대해, 그룹 관리서버는 그룹키를 갱신하고 갱신된 그룹키를 모든 멤버에게 안전한 유니캐스트 채널을 통해 전달한다. 그러므로 그룹서버는 각 멤버에게 키 갱신 메시지를 안전하게 전달하기 위해 n번의 암호화를 수행해야 한다. 각 멤버는 그룹키와 그룹 관리서버와 공유하는 비밀키만을 가지므로 멤버는 두 개의 키를 저장하게 되고 그룹 관리서버는 그룹키와 n명의 멤버와의 비밀키를 가지므로 $n+1$개의 키를 저장한다.



[그림 2] 단순 그룹키 공유 방식

이 방법은 구현이 쉽고 간단하며 키 관리를 위한 부가적인 구조를 필요로 하지 않는 장점을 가지지만 멀티캐스트 그룹의 크기와 동적인 성질에 대해 확장성을 제공하지 못 하며, 멤버의 참여와 탈퇴시 키 갱신을 위한 그룹관리 서버의 작업량이 그룹 멤버의 수에 대해 선형적으로 증가하는 단점을 가진다. 그러므로 이 방법은

그룹의 크기가 작은 특수한 경우에 사용될 수 있으며 안전한 멀티캐스트를 위한 그룹 통신에는 부적절하다.

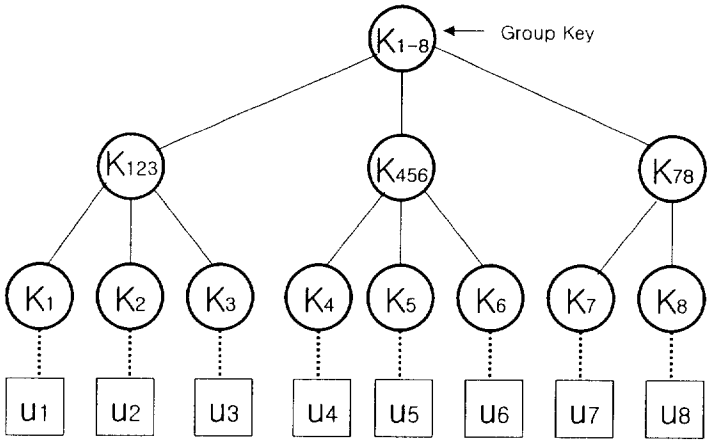
3.2 트리 기반의 그룹키 관리 기법

트리 기반의 키 관리는 n명의 멤버에 대해 n개의 단말노드(leaf node)를 가지며 트리의 깊이가 $\log_k(n)$인 k차 트리를 구성하고, 각 단말노드를 멤버와 그룹관리자와 공유하는 비밀키로 하고 중간 노드들을 키 갱신을 위한 키 암호화 키(key encryption key)로 사용하는 방법이다. 그러므로 멤버들이 가지게 되는 키의 개수와 키 갱신에 필요한 메시지가 $O(\log_k(n))$이 된다.

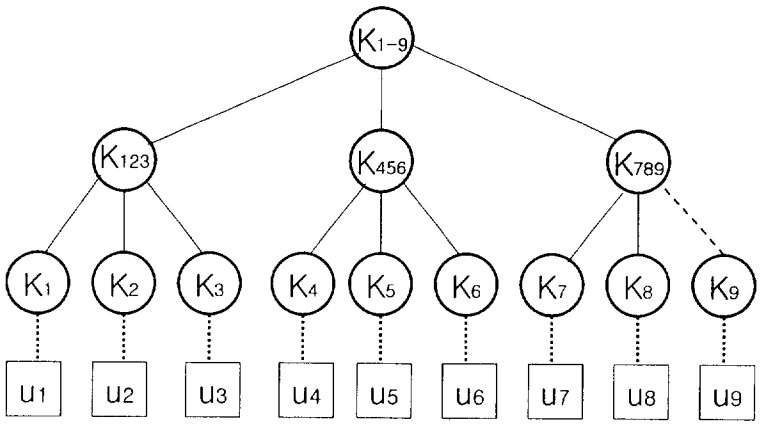
3.2.1 키 트리를 이용한 그룹키 관리

그룹키 관리의 확장성을 제공하고 키 갱신에 대한 비용을 줄이기 위해 [6][11]에서는 키 트리(key tree)라는 키 암호화키(key encrypting key)의 논리적인 계층구조를 사용하였다. 트리 기반의 그룹키 관리 기법에서 멀티캐스트 그룹은 u_1에서 u_n으로 표기되는 n명의 그룹 멤버와 그룹 관리를 담당하는 중앙의 그룹서버로 구성된다. 그룹서버는 각 노드가 키로 구성되는 k차(k-ary) 트리를 구성하며, 이때 단말노드들은 n명의 멤버들에 대한 비밀키로 구성되며 멤버 u_j가 저장하는 그룹서버의 키들의 부분집합은 키 그래프에서 u_j에 해당하는 단말노드 j에서 트리의 루트노드까지의 경로상의 키들의 집합과 같다. 예를 들면, 그림 3에서 멤버 u_6이 소유하게 되는 키들의 집합은 키 그래프의 경로 상에 있는 $\{K_{1-8}, K_{456}, K_6\}$이다.

키 그래프에서 루트노드의 키 K_{18}은 그룹의 공유키가 되고 중간 노드들은 키 갱신을 위한 보조키들(auxiliary keys) 또는 키 암호화키들(key encryption keys)로 사용된다.



[그림 3] n=8, k=3인 키 트리



[그림 4] u_9의 그룹 참여시 키 트리

1) 그룹 참여(join)

어떤 사용자 u_i의 그룹 참여가 허용되면 그룹관리서버 GS는 u_i의 비밀키 K_i를 가지는 새로운 노드를 생성하고 K_i노드가 추가될 참여지점(joining point)을 찾아 K_i를 해당 노드의 자식 노드로 둔다. 그림 3과 그림 4에서 u_9가 그룹에 참여하는 경우, K_{78}이 참여지점이 되며 u_9의 비밀키를 가지는 K_9노드를 K_{789}의 자식 노드로 추가한다.

Backward secrecy를 위해 참여지점인 K_{78}는 새로운 키 K_{789}로 갱신되고 그룹키 K_{1-8}은 K_{1-9}로 갱신된다. 갱신된 키에 대한 분배는 user-oriented와 key-oriented 방식을 통해 분배될 수 있다.

i) User-oriented rekeying

user-oriented 키 갱신은 갱신된 새로운 키를 필요로 하는 사용자들의 집합으로 키를 전송하는 방법이다. 그림 4에서 u_9의 참여에 대한 키 분배는 다음과 같이 수행된다. $\{\}_{K_i}$는 {}가 키 K_i로 암호화됨을 의미한다.

$$GS \Rightarrow \{u_1, \ldots, u_6\} \quad : \{K_{1-9}\}_{K_{1-8}}$$

$$GS \Rightarrow \{u_7, u_8\} \quad : \{K_{1-9},\ K_{789}\}_{K_{78}}$$

$$GS \Rightarrow u_9 \quad : \{K_{1-9},\ K_{789}\}_{K_9}$$

ii) Key-oriented rekeying

key-oriented 키 갱신은 새로운 키를 개별적으로 암호화하는 방식으로 이에 대한 키 분배는 아래와 같이 수행된다.

$GS \Rightarrow \{u_1, \ldots, u_8\}$: $\{K_{1-9}\}_{K_{1-8}}$

$GS \Rightarrow \{u_7,\ u_8\}$: $\{K_{1-9}\}_{K_{1-8}}$, $\{K_{789}\}_{K_{78}}$

$GS \Rightarrow u_9$: $\{K_{1-9},\ K_{789}\}_{K_9}$

2) 그룹 탈퇴(leave)

어떤 멤버가 탈퇴하는 경우, 탈퇴하는 멤버에 대한 forward secrecy를 위해 탈퇴하는 멤버가 알고있는 모든 키들이 갱신되어야 한다. 만약 멤버 u_9가 탈퇴한다면 u_9가 소유했던 키 집합 $\{K_{1-9}, K_{789}\}$을 $\{K_{1-8}, K_{78}\}$로 갱신하고 K_9를 키 그래프에서 삭제하며 남아있는 모든 멤버들에게 갱신된 키들을 안전하게 분배해야 한다.

i) User-oriented rekeying

u_9의 탈퇴로 인한 키의 갱신은 다음과 같이 수행된다.

$GS \Rightarrow \{u_1,\ u_2,\ u_3\}$: $\{K_{1-8}\}_{K_{123}}$

$GS \Rightarrow \{u_4,\ u_5,\ u_6\}$: $\{K_{1-8}\}_{K_{456}}$

$GS \Rightarrow u_7$: $\{K_{1-8},\ K_{78}\}_{K_7}$

$GS \Rightarrow u_8$: $\{K_{1-8},\ K_{78}\}_{K_8}$

ii) Key-oriented rekeying

$GS \Rightarrow \{u_1,\ u_2,\ u_3\}$: $\{K_{1-8}\}_{K_{123}}$

$GS \Rightarrow \{u_4,\ u_5,\ u_6\}$: $\{K_{1-8}\}_{K_{456}}$

$GS \Rightarrow u_7$: $\{K_{1-8}\}_{K_{78}}$, $\{K_{78}\}_{K_7}$

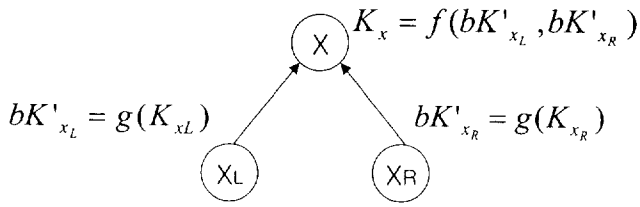
$GS \Rightarrow u_8$: $\{K_{1-8}\}_{K_{78}}$, $\{K_{78}\}_{K_8}$

3.2.2 One-way Function Trees

One-way Function Tree(OFT)[4]는 멤버의 참여나 탈퇴시 키 갱신을 위한 그룹 관리서버의 계산량을 줄이기 위해 키 트리 기법의 변형된 방법을 제안하였다.

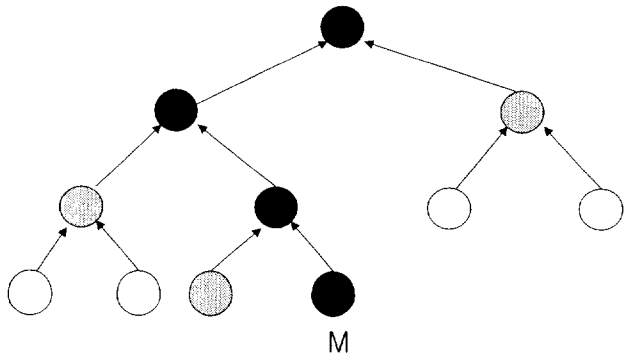
이 방법은 키 트리에서 새로운 그룹키를 계산하기 위해 일방향함수(one-way function)를 키 트리에 상향식(bottom-up)으로 적용하였다. 중앙의 그룹관리서버는 일반적인 키 트리 기법처럼 사용자의 키를 단말노드로 가지고 루트노드가 그룹키가 되는 이진 키 트리(binary key tree)를 관리한다.

키 트리에서 각 노드 x는 노드 키(node key) K_x와 blinded node key라는 bK'_x 두개의 키와 연관되어진다. 이때 blinded node key $bK'_x = g(K_x)$이며, 함수 $g()$는 일방향함수이다. 즉, blinded key는 실제 키의 값을 직접 드러내지 않도록 숨기는 역할을 하게된다. 함수 g가 일방향함수이므로 blinded node key bK'_x로부터 노드 키 K_x를 구하는 것은 계산상 불가능하다는 성질을 가지게 된다. 키 트리의 구조에서 중간 노드들은 모두 두개의 자식 노드를 가지며 단말노드는 각 사용자의 비밀키로 구성되고 내부노드 x의 키 K_x는 다음 식에 의해 계산되고 이를 도식화하면 그림 5와 같다. 여기서 x_L과 x_R은 노드 x의 왼쪽과 오른쪽 자식노드를 나타낸다. 함수 f는 혼합함수(mixing function)로써 일방향함수일 필요는 없다.



[그림 5] OFT의 노드키 계산

그룹 멤버는 자신에 해당하는 단말노드의 키와 루트노드까지의 경로상의 형제 노드들(sibling nodes)의 blinded key들을 관리하고, 함수 g와 f를 사용해서 경로상의 노드의 노드키와 루트키인 그룹키를 계산해낼 수 있다. 그림 6은 OFT 트리의 예제이다. M으로 표기된 단말노드에 해당하는 멤버는 빗금으로 표시된 노드의 blinded 키들만 알고 있고 자신의 비밀키와 blinded 키들을 함수 $g()$와 $f()$를 이용해서 경로상의 노드들의 키를 계산하고 최종적으로 루트노드의 그룹키를 계산할 수 있다.

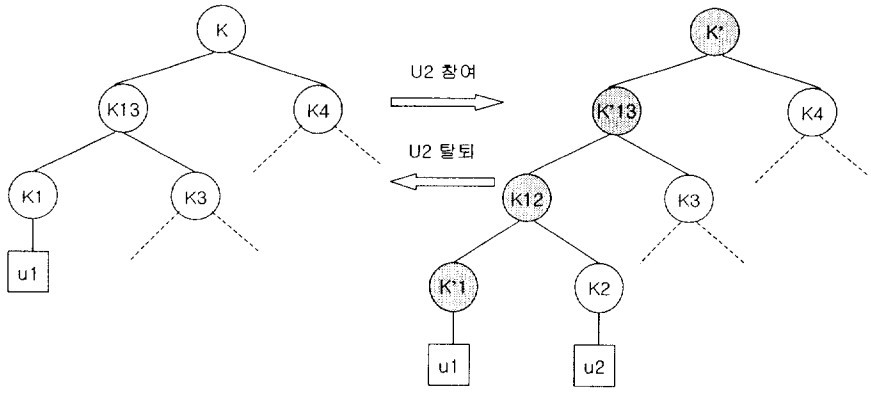


[그림 6] OFT 트리의 구성

어떤 멤버가 그룹에 참여하거나 탈퇴할 때, 해당 멤버에 대한 트리의 경로상의 모든 키들이 갱신되어야 한다. 하지만 내부 노드의 키들은 다른 키들에 의해 계산

될 수 있으므로 그룹서버는 해당 멤버에 의해 영향 받는 단말노드의 키 하나만 새로 생성하거나 갱신하면 된다. 그리고 그룹서버는 변경된 키를 이용해서 내부노드들의 키를 계산하고 그룹내의 모든 멤버들이 멤버십의 변화로 인해 변경된 키들을 다시 계산할 수 있도록 새롭게 계산된 키들에 대한 정보를 안전하게 멀티캐스트 해야 한다. 그룹 멤버의 수가 n명이라 할 때, 그룹서버는 키 갱신을 위해 $\log_2 n$개의 메시지를 필요로 하며, 실제로 전송되는 메시지는 멤버십이 변경되는 해당 멤버에서 루트노드까지의 경로상의 모든 형제노드들에 대한 블라인드 키들이다.

OFT 기법은 멤버의 추가와 삭제로 인한 키 갱신의 수를 최소화하기 위해 키 트리를 full-balanced형태로 유지하게 되며, 키 갱신에 필요한 메시지의 수를 $\log_2 n$으로 감소시켰다. 일반적인 키 트리 방식에서 키 갱신이 그룹서버에서만 계산되고 그룹 멤버에 의한 계산이 필요 없는 반면 OFT는 하위 레벨의 노드 키들과 blind 키들로부터 상위 레벨의 키를 유도하기 위해 그룹멤버의 계산 능력을 필요로 한다. 그러므로 OFT 기법은 그룹 멤버가 상위 레벨의 키를 계산하는 과정에서 $2\log_2 n$ 개의 키들을 위한 저장공간을 요구한다.



[그림 7] OFT의 멤버 참여/탈퇴에 대한 키 트리 갱신

1) 그룹 참여(join)

어떤 멤버 u가 그룹에 새로 참여하는 경우, 그룹 관리자는 u에 연관되는 노드 n을 생성하고 n에 임의로 선택한 u와의 비밀키를 할당하고 u에게 안전한 통신 채널을 통해 이 키를 전달한다. Full-balanced 트리를 유지하기 위해 u에 대한 노드 n은 가장 얕은 단말노드(shallowest leaf node)를 분할하여 오른쪽 자식 노드로 연결되며 왼쪽 자식노드는 기존의 단말노드가 연결된다. 그림 7은 u_2가 새로 그룹에 참여한 경우의 예를 보여준다.

그림 7에서 가장 얕은 단말노드는 K_1이 되며, K_1은 새로운 노드 K_{12}로 대체되고 왼쪽 자식노드로 u_1이 연결되고 새로운 키 K'_1이 할당되고, u_2는 오른쪽 자식노드로 연결되며 키 K_2가 할당된다. 그룹 관리자는 u_1과 u_2에게 각각 새로 할당된 키를 전달하고 다음의 처리를 수행한다.

$$GM \Rightarrow group \quad : \{K'_1, bK_2\}_{K_1}, \{bK_{12}\}_{K_3}, \{bK'_{13}\}_{K_4}$$

$$GM \rightarrow u_2 \quad : \{bK_{1'}, bK_{3'}, bK_4'\}_{K_2}$$

여기서 BK_i는 i노드의 키 K_i에 일방향함수가 적용된 blinded key이다 모든 그룹 멤버들은 위의 메시지를 수신하게 되고, 다음 계산을 통해 새로운 노드 키들을 계산하게 된다.

$$K_{12} = f(g(K'_1), g(K_2))$$

$$K'_{13} = f(g(K_{12}), g(K_3))$$

$$K' = f(g(K'_{13}), g(K_4))$$

2) 그룹 탈퇴(leave)

어떤 멤버 u가 그룹을 탈퇴하거나 그룹에서 제거되는 경우, u 노드의 형제노드(sibling node) s가 u의 부모노드에 대체되고 그룹 관리자는 s에 새로운 키를 할당한다. 그러므로 s에서 루트노드까지의 경로상의 모든 키들이 일방향함수와 혼합함수에 의해 갱신되고 갱신된 키들은 남아있는 멤버들에게 암호화되어 분배된다. 탈퇴한 멤버 u의 키 값이 그룹키 계산에 이용되지 않았으므로 u는 더 이상 그룹키에 접근하지 못 하게된다. 그림 7에서 u_2가 트리에서 제거되는 경우, K'_1이 K_{12}의 위치로 이동하고 u_1에 대한 새로운 키 K_1이 할당되고 그룹 관리자는 다음의 처리를 수행하며, 그러므로 모든 그룹 멤버들은 새로운 키들을 계산할 수 있다.

$$GM \Rightarrow group : \{K_1\}_{K'_1}, \{bK_1\}_{K_3}, \{bK_{13}\}_{K_4}$$

3) OFT의 특성

OFT에서 키에 대한 안전성은 키 트리에서 사용되는 일방향함수에 의존하게 되며, 그러므로 그룹관리서버는 새로운 멤버의 추가시 단지 하나의 비밀키만 생성하면 된다. 또한 그룹에서 제거된 멤버가 기존의 경로상의 sibling node의 blinded key들을 알고 있다고 하더라도 비밀키들을 계산하기에 충분하지 않으므로 갱신된 그룹키를 계산할 수 없다

OFT는 기존의 트리 기반의 방식[11]에 비해 멤버의 추가와 삭제로 인한 키 갱신을 위해 적은 양의 메시지를 기존의 멤버들에게 멀티캐스트 하게 된다. 즉, 멤버

들에게 일방향함수에 대한 계산능력을 부가함으로써 키 갱신에 필요한 메시지의 수를 감소시켰다. 표 3은 OFT에서의 계산량과 메모리 요구사항을 비교해서 나타내고 있다.

[표 3] OFT의 키 갱신에 대한 계산량 비교

	키 공유 방식	키 트리 기반	OFT
유형	중앙집중식	중앙집중식	중앙집중식
서버의 관리 키 수	$n+1$	$(k^n-1)/(k-1)$	$2n$
멤버의 관리 키 수	1	$\log_k(n)$	$2\log_2 n$
키 갱신 메시지 수	n	$k\log_k(n)$	$\log_2(n)$
join/leave 서버 계산량	$O(n)$	$O(k\log(n))$	$O(\log n)$
join/leave 멤버 계산량	$O(1)$	$O(\log(n))$	$O(\log n)$
Forward secrecy	Yes	Yes	Yes
Backward secrecy	Yes	Yes	Yes

n : Group size(멤버의 수), k : 키 트리의 차수

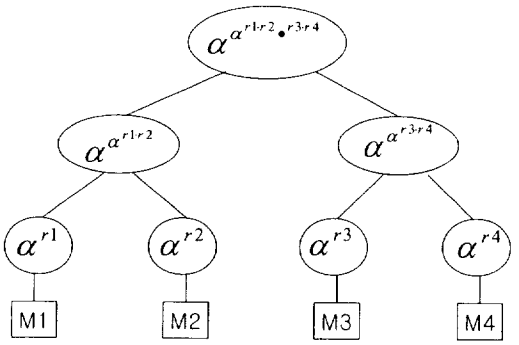
3.3 GDH(Group Diffie-Hellman)을 이용한 그룹키 관리

3.3.1 Authenticated Group Key Agreement(AGKA) protocol

AGKA[12]는 DH 키 교환 프로토콜을 그룹으로 확장한 것으로 일반적인 DH 프로토콜을 그대로 그룹키에 적용한 NAGKA(Non Authenticated Group Key Agreement)와 인증 기능을 포함한 AGKA(Authenticated Group Key Agreement)로 구분된다. 각 멤버들이 그룹키 계산을 담당하므로 중앙의 키 관리서버의 존재를 가정하지 않는다.

1) NAGKA

NAGKA는 그룹멤버들의 익명성을 제공해주기 위한 환경에서 사용될 수 있는 프로토콜로서 2-party의 DH 프로토콜을 group key agreement로 확장한 방법이다. 그룹에 대한 키 트리를 구성하고 각 멤버들은 자신의 단말노드에 대한 비밀값 r_i를 생성하고, 첫 라운드에서 서브트리의 특정 두 멤버가 키 협정을 통해 상위 노드의 키를 계산하고 계산된 키를 그룹으로 브로드캐스트 한다. 그리고 다음 단계의 상위 노드의 키를 계산하기 위해 각 서브트리의 두 멤버가 다시 키 협정을 수행한다. 이러한 동작을 반복함으로써 루트 노드의 그룹키를 계산하게 된다. 이때 각 단계에서의 키 협정은 서브트리에서 제일 왼쪽 노드의 멤버가 담당한다. 그림 8은 NAGKA의 트리 구성을 보여준다. 이 방식은 인증 기능은 제공하지 않으므로 기존의 DH 프로토콜의 문제점인 man-in-the-middle-attack이 가능하며, 그룹의 기밀성을 해치게된다.



[그림 8] NAGKA의 그룹키 계산

노드 (l, v)의 키 $K_{(l,v)}$의 계산은 다음과 같이 수행된다.

$M_L \Rightarrow group$: $\alpha^{K_{(l+1,v_l)}} \bmod p$; M_L : left subgroup member

$$M_R \Rightarrow group \quad : \quad \alpha^{K_{(l+1,v_R)}} \bmod p \quad ; \; M_R : \text{right subgroup member}$$

$$\begin{aligned} K_{(l,v)} &= (\alpha^{K_{(l+1,v_L)}})^{K_{(l+1,v_R)}} \bmod p \\ &= (\alpha^{K_{(l+1,v_R)}})^{K_{(l+1,v_L)}} \bmod p \\ &= \alpha^{K_{(l+1,v_L)} \cdot K_{(l+1,v_R)}} \bmod p \end{aligned}$$

노드키 $K_{(l,v)}$의 계산에서 $(l+1, v_L)$은 노드 (l, v)의 왼쪽 자식노드를 나타내고 $(l+1, v_R)$은 오른쪽 자식노드를 나타낸다.

2) AGKA

키 협정에서의 인증을 위해 PKI 기반의 인증서(certificate)를 사용할 수 있으나 크기가 큰 인증서의 잦은 교환은 대역폭을 소모할 뿐만 아니라 서명의 검증은 상당한 계산량을 필요로 하므로, AGKA-G[12]에서는 KAC(key authentication center)라는 신뢰되는 기관을 통해 인증서 없이 키 협정에서 인증 기능을 수행하는 방안을 제안하였다. 이 방법은 [13]의 identity based key agreement protocol을 기반으로 하고 있으며, 두 서브트리 멤버간의 유니캐스트 채널을 통한 상호인증(mutual authentication)을 통해 그룹키를 생성하게 된다. Procedure 3-1과 Procedure 3-2는 pre-authentication과 키 협정을 나타낸다.

Initialization:

KAC는

소수 p를 생성하고 Z_p^*상의 생성자 α를 선택.

비밀값 $x \in Z_p^*$ 선택, $y = \alpha^x \bmod p$와 p를 공개값으로 설정

Preauthentication:

A는 KAC에 자신의 식별자 ID_A를 등록

$KAC \rightarrow A$

임의의 값 K_A를 선택, 여기서 $\gcd(K_A, p-1) = 1$

$P_A = \alpha^{K_A} \bmod p$, $K_A^{-1} \bmod (p-1)$ 계산

$a = K_A^{-1} \cdot (H(ID_A) - x \cdot P_A) \bmod (p-1)$; H : hash function

a는 A의 비밀값이 되고 P_A와 ID_A는 공개

[Procedure 1] Pre-authentication

Key agreement:

1. $A \rightarrow B$: ID_A , P_A
2. $B \rightarrow A$: ID_B, P_B
3. $A \rightarrow B$: $(P_B)^{r_a} \bmod p$; r_a는 A의 비밀값
4. $B \rightarrow A$: $(P_A)^{r_b} \bmod p$; r_b는 B의 비밀값
5. 키 계산

$K = \alpha^{K_A \cdot a \cdot r_b + K_B \cdot b \cdot r_a} \bmod p$

[Protocol 1] 두 멤버간의 Key agreement

위의 프로토콜에서 A는 다음의 계산과정을 통해 협정된 키 K를 계산할 수 있다.

$\alpha^{K_A \cdot a \cdot r_B}$는 획득한 파라미터로부터 계산

$$\alpha^{K_B \cdot b} \bmod p = \alpha^{H(ID_B) - x \cdot P_b} \bmod p = \frac{\alpha^{H(ID_B)}}{(\alpha^x)^{P_B}} \bmod p$$

$$\alpha^{K_B \cdot b \cdot r_a} = \left(\frac{\alpha^{H(ID_B)}}{(\alpha^x)^{P_B}}\right)^{r_a} \bmod p$$

즉, A는 키 계산을 위해 공개값 I_B와 A의 비밀값 r_A를 이용

각 단계에서 키 계산을 담당하는 서브트리의 멤버들은 위의 프로토콜을 반복적으로 수행함으로써 상위노드의 키를 계산할 수 있게 된다.

3) AGKA의 특성

AGKA는 작은 규모의 그룹을 위한 키 관리에 적합하도록 설계되었으며, 그룹내의 모든 멤버간의 멤버 대 멤버(member-to-member)에 대한 인증 없이 상호 인증을 수행할 수 있다. 표 4는 N명의 멤버에 대한 AGKA의 통신상의 비용과 계산상의 비용을 나타낸다.

[표 4] NAGKA와 AGKA 계산량

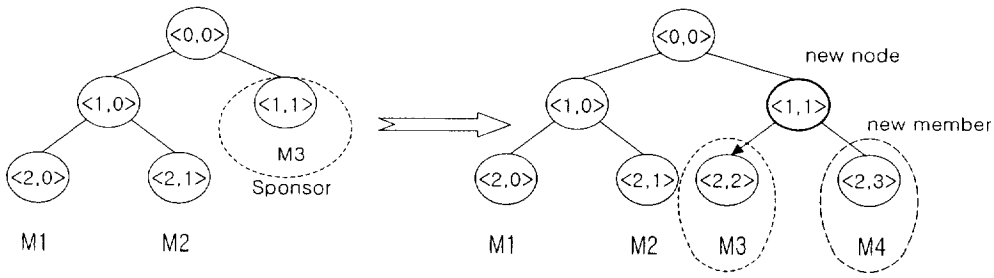
	NGKA	AGKA
rounds	$d = \lceil \log(N) \rceil$	$d = \lceil \log(N) \rceil$
broadcasts	2(N-1)	2(N-1)
total msgs	2(N-1)	8(N-1)
total bandwidth	2(N-1)|K|	8(N-1)|K|
exp per M_i(min, max)	d+1, 2d	5, 5d
exp total	(d+2)N	10(N-1)

3.3.2 TGDH(Tree-base Group Diffie-Hellman) protocol

TGDH[9]는 AGKA와 유사하나 두 멤버간의 상호인증을 통해 그룹키를 계산하는 것이 아니라, 그룹의 멤버십의 변경으로 인한 키의 관리를 해당 세션에서의 스폰서(sponsor)라는 특정 그룹 멤버가 담당하도록 하고 있다. TGDH에서도 노드의 secret key와 blinded key를 사용하나 TGDH에서 blinded key $bK = f(K) = \alpha^K \pmod{P}$이다.

1) Join Protocol

멀티캐스트 그룹에 n명의 멤버 $\{M_1, M_2, \ldots\ldots, M_n\}$가 있다고 할 때, 새로 참여하기를 원하는 멤버 M_{n+1}은 그룹 참여 요구 메시지와 함께 자신의 blinded key $BK_{\langle l,v \rangle}$를 그룹에 전송한다. 현재 트리에서 가장 오른쪽의 낮은 노드의 그룹 멤버가 스폰서가 되며, 스폰서는 새로운 멤버에 대한 노드를 추가하고 새로운 그룹키를 계산한 후 모든 blinded 키를 그룹으로 broadcast 전송한다. 그림 9는 멤버 M_4가 추가되는 과정을 보여준다.



[그림 9] 멤버 참여에 대한 키 트리 갱신

그림 9에서 M_3가 스폰서가 되며 다음 작업을 수행한다.

(1) 노드 $\langle 1,1\rangle$을 $\langle 2,2\rangle$로 변경

(2) 새로운 중간노드 $\langle 1,1\rangle$과 새로운 멤버노드 $\langle 2,3\rangle$을 생성

(3) 노드 $\langle 1,1\rangle$을 $\langle 2,2\rangle$와 $\langle 2,3\rangle$의 부모 노드로 설정

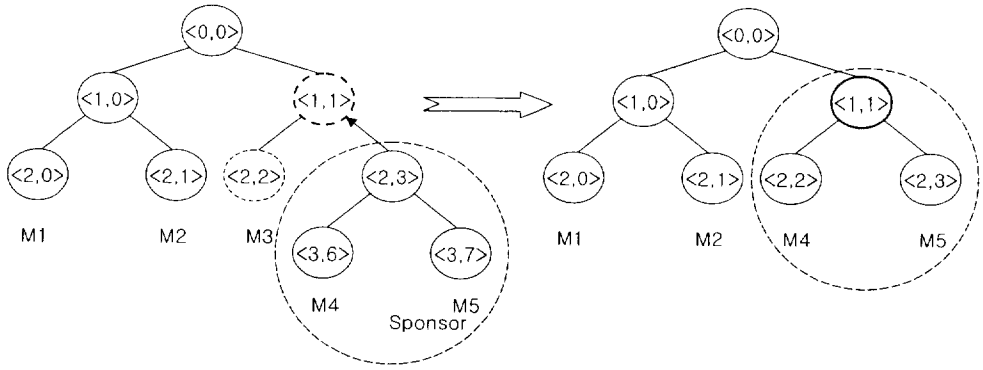
(4) $bK_{\langle 2,3\rangle}$과 $bK_{\langle 1,0\rangle}$를 이용해서 그룹키 $K_{\langle 0,0\rangle}$를 계산

(5) 모든 blinded 키들을 그룹으로 전송

기존의 다른 멤버들도 (1)과 (2)를 통해 트리를 갱신하지만 아직 새로운 그룹키를 계산할 수는 없다. 그룹의 다른 멤버들은 스폰서에 의해 전송된 blinded 키들을 수신한 다음 그룹키를 계산할 수 있게 된다.

2) Leave Protocol

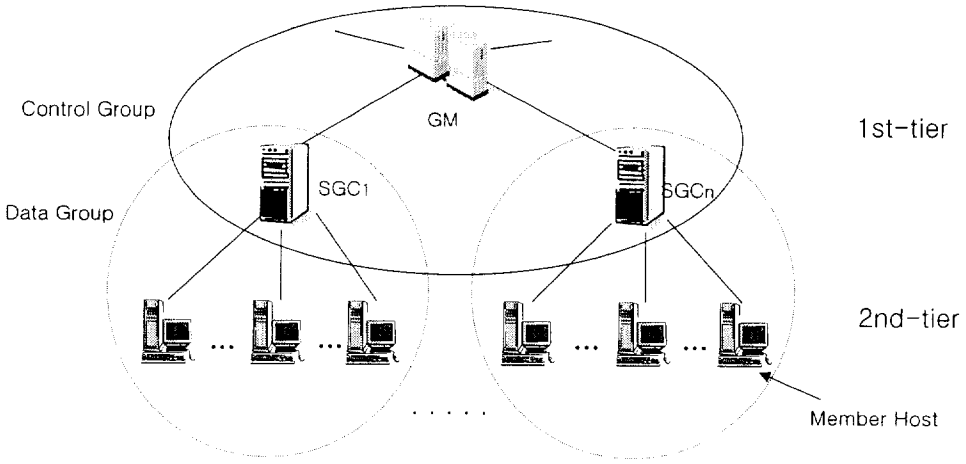
어떤 멤버 M_d가 탈퇴하는 경우, 키 트리에서 탈퇴하는 멤버의 형제노드(sibling node)를 루트노드로 하는 서브트리의 가장 오른쪽의 단말노드가 스폰서가 된다. 탈퇴 프로토콜에서 모든 멤버들은 M_d에 대응되는 단말노드를 삭제함으로써 자신의 키 트리를 갱신하게 되며, M_d 노드의 형제노드가 M_d노드의 부모노드의 위치로 승격된다. 스폰서는 키 트리 경로상의 모든 키들을 새로 계산하고 모든 blinded 키들을 그룹으로 전송하며, 모든 멤버들은 이 정보를 이용해서 새로운 그룹키를 계산하게 된다. 그림 10은 멤버 M_3의 탈퇴시 키 트리의 갱신과정을 보여준다. 멤버 M_3가 탈퇴하는 경우, 스폰서는 M_5가 되며 키트리에서 노드 $\langle 1,1\rangle$과 $\langle 2,3\rangle$을 삭제함으로써 트리를 갱신한다. 그리고 M_5는 키 $K_{\langle 2,3\rangle}$를 새로 생성하고 $K_{\langle 1,1\rangle}$, $K_{\langle 1,0\rangle}$, $bK_{\langle 2,3\rangle}$ 그리고 $bK_{\langle 1,1\rangle}$를 계산하여 그룹으로 전송한다. M_3가 모든 blinded 키들을 알고있다 하더라도 자신의 공유값이 그룹키의 계산에 사용되지 않았으므로 새로운 그룹키를 계산할 수 없다.



[그림 10] 멤버 탈퇴에 대한 키 트리 갱신

4. 확장성을 위한 그룹키 관리 구조 방안

4.1 2-계층(2-tier) 방식의 그룹관리 구조



[그림 11] 2-계층 형태의 그룹키 관리 구조

본 논문에서는 전반적인 그룹구조를 그림 11 형태의 2-계층(2-tier)으로 구성한다. 첫 번째 계층(1st tier)은 하부의 subgroup controller(SGCs)들에 대한 계층으로 GM의 관리 영역이다. 두 번째 계층(2nd-tier)은 실제 멤버 호스트들로 구성되는 서브그룹으로, 상대적으로 적은 수의 멤버들로 구성되는 소규모의 그룹들로 구성되며 각 서브그룹은 해당 서브그룹 관리자 SGC의 관리 영역이다. 그리고 각 서브그룹은 전체 그룹에서 자신들만의 자치(autonomous) 그룹을 형성하고 각 서브그룹 관리자 SGC가 해당 그룹에 대한 관리권한을 가지게 된다.

하부계층의 서브그룹은 물리적인 형태에 따라 지리적으로 밀집된 멤버들의 그룹으로 구성되는 집합체로 볼 수도 있고, 데이터의 동질성에 따라 멤버들의 집합으로 구성되는 논리적인 구조로 볼 수도 있다. 데이터의 동질성에 의해 서브그룹을

구성하는 경우 서브그룹간에는 전체 그룹내에서의 상하계층관계가 발생할 수 있으며, 상위계층의 멤버는 하위계층의 데이터에 접근할 수 있으나 하위계층의 멤버는 상위계층의 데이터에 접근할 수 없는 계층그룹간의 접근제어도 제공해야 한다[14].

전반적인 그룹 구조를 계층화시킴으로써 group manager의 workload를 각 subgroup controller들에게 분담하고 그룹관리의 확장성(scalability)을 제공한다. GM은 전반적인 그룹에 대한 정책결정 및 그룹에 참여하고자 하는 멤버에 대한 pre-authentication만을 수행하며, 전체 멤버들을 직접 관리하지 않고 대신에 SGC들로 구성되는 상부계층을 관리하게 된다. SGC는 해당 서브그룹에 대한 관리를 책임지며 다른 그룹으로 데이터 전송에 대한 중재자(intermediary)역할을 한다. 참여하는 그룹멤버에 대한 키의 분배는 서브그룹 수준에서 해당 SGC가 수행하게되므로 그룹 멤버의 참여와 탈퇴로 인한 키의 갱신은 해당 서브그룹에서만 발생하게 되고, 어떤 서브그룹에서의 멤버십의 변경이 다른 그룹에 영향을 주지 않는다.

상부계층에서의 안전한 통신을 위해 모든 SGC들은 상부계층의 그룹관리 메커니즘을 통해 control group key CGK를 공유하며, 각 서브그룹 SG_i에서는 SGC_i와 멤버 M_i^*들간의 서브그룹키 SGK_i를 공유함으로써 서브그룹내에서의 안전한 통신을 수행하게 된다. 각 서브그룹은 독자적인 멀티캐스트 주소와 그룹관리 기법을 사용할 수 있으며 또한 서브그룹의 키는 서로 독립적으로 사용될 수 있다. 상부계층(1st-tier)의 키 관리는 decentralize 방식을 사용함으로써 Iolus에서 제기되었던 중앙 관리서버의 오류로 인한 one-point failure문제를 방지한다. 그리고 각 중앙의 그룹관리자와 서브그룹 관리자들간의 그룹 관리 메시지를 교환함으로써 그룹 관리에 대한 작업을 분담하도록 한다.

본 논문에서 초기 시스템 설정단계에서는 SGC들의 AGKA(Authenticated Group Key Agreement)를 사용하고 SGC들의 참여와 탈퇴로 인한 키의 갱신은 TGDH를 가정한다. 그리고 각 하부계층(2nd-tier)의 서브그룹의 키 관리는 중앙집중식 형태의 OFT 기법을 사용하도록 하며 GM과 SGCs들간의 신뢰성을 가정한다.

4.2 그룹 운용(Group operation) 방식

4.2.1 그룹 초기화(Group Initialize)

그룹의 초기화는 상부계층에 대한 초기화와 각 서브그룹에 대한 하부계층의 초기화로 구별된다. GM은 implicitly certified public key(ICPK)를 위한 파라미터를 생성하고 공개하며, 각 SGC_i의 AGKA를 위한 ICPK를 제공한다. 본 논문에서는 ICPK의 생성을 위해 Zheng의 SDSS(Shortened DSS) 알고리즘을 변형하여 사용함으로써 [13]에서의 법-역원의 계산을 제거함으로써 계산상의 효율성을 증가시켰다.

Round 1:

GM

chooses prime p, q and generator $\alpha \in Z_q^*$; (q divides $p-1$)

selects secret $x \in Z_q^*$ and computes $y = \alpha^x \bmod p$

p, q, α, y are published

for each SGC_i , distributes ICPK

selects key K_i and K_i^{-1}

;where $\gcd(K_i , p-1)=1$ and $K_i \cdot K_i^{-1} \equiv 1 (\bmod p-1)$

$P_i = \alpha^{K_i} \bmod p$

$S_i = K_i^{-1} \cdot (hash(ID_i , P_i) + x) \bmod q$

ID_i , S_i , P_i

Each SGC_i runs AGKA and calculates control group key

[Procedure 2] GM과 SGC의 그룹 초기화

상부계층의 초기화를 위해 각 SGC_i는 AGKA 프로토콜에 따라 상부계층의 그룹키 CGK를 생성함으로써 상부계층을 구성한다. 그룹 초기화는 자신의 서브그룹

에 멤버가 존재하는 SGC만이 참여하며, 멤버가 존재하지 않는 SGC는 초기화 단계에서 그룹에 참여할 필요가 없으며 이후에 필요에 따라 그룹에 참여하며, SGC의 그룹 참여로 인한 상부계층의 키 갱신이 발생한다.

상부계층이 구성되고 나면 각 SGC_i는 자신의 서브그룹을 구성하기 위해 해당 서브그룹의 멤버들에 대한 OFT 키트리를 구성하고 멤버들에게 서브그룹키 SGK_i를 전달한다.

Round 2:
  for each SGC_i
    constructs OFT key tree for each subgroup SG_i
    calculates subgroup key SGK_i
    distributes SGK_i

[Procedure 3] 각 SGC의 서브그룹 초기화

4.2.2 Join Protocol

어떤 사용자 u가 그룹에 참여하고자 하는 경우 u는 GM의 pre-authentication을 통해 ICPK를 부여받고, 자신이 소속되는 서브그룹으로 참여한다. 이 때 GM과의 pre-authentication을 통한 ICPK의 분배는 SSL과 같은 안전한 통신 채널을 사용하도록 한다.

Pre-authentication을 수행한 u는 자신의 공개키 P_u와 식별자 ID_u를 자신이 소속하게 되는 서브그룹의 관리자 SGC_i에게 전달하며, SGC_i는 u에 대한 key agreement 프로토콜을 수행하고 SGC_i의 서브그룹의 멤버로 참여시킨다. 실제 구

현에 있어서 새로운 멤버가 참여하게 되는 서브그룹을 GM이 지정해 줄 수도 있고, 또는 u가 자신의 참여 정보를 그룹으로 broadcast하고 이에 대한 응답이 먼저 도착하는 SGC와 key agreement를 통해 해당 서브그룹으로 참여할 수도 있다.

$u \rightarrow GM$: $join\_request$

GM :

selects unique ID ID_u

computes K_u , $P_u = \alpha^{K_u} \bmod p$ and secret S_u

; $S_u = K_u^{-1} \cdot (hash(ID_u , P_u) + x) \bmod q$

$GM \rightarrow u$: ID_u , P_u , S_u

[Protocol 2] GM과 사용자의 pre-authentication

$u \Rightarrow group$: ID_u , P_u

$SGC_i \rightarrow u$: ID_i , P_i

u and SGC_i choose secret r_u and r_i respectively

$u \rightarrow SGC_i$: $(P_i)^{r_u} \bmod p$

$SGC_i \rightarrow u$: $(P_u)^{r_i} \bmod p$

u and SGC_i compute key $K = \alpha^{K_u \cdot S_u \cdot r_i + K_i \cdot S_i \cdot r_u} \bmod p$

[Protocol 3] 그룹 참여시 SGC와 멤버간의 key agreement

Key agreement의 성공적인 수행 여부를 확인하기 위해 u와 SGC_i는 각각 자신의 식별자를 키 K로 암호화해서 교환한다.

$u \rightarrow SGC_i$: $\{ID_u\}_K$

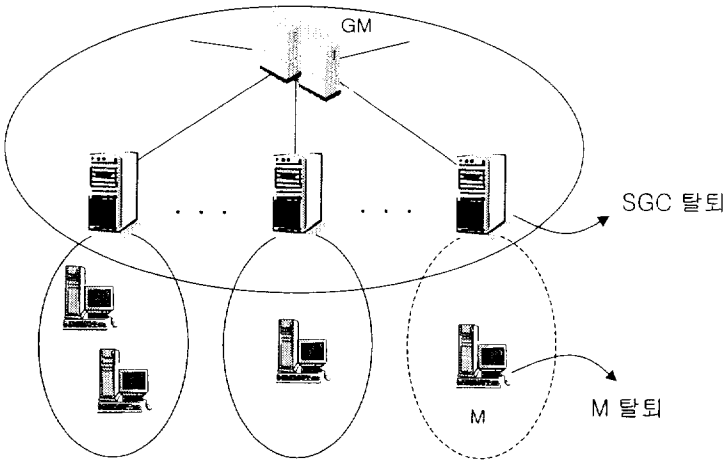
$SGC_i \rightarrow u$: $\{ID_i\}_K$, $\{BlindedKeys\}_K$

SGC_i는 복호화를 통해 u의 식별자 I_u를 확인하면 u를 멤버로 참여시키고, 새로운 멤버 u에 대한 노드의 키를 key agreement를 통해 생성된 키 K로 할당하고 OFT 프로토콜에 의해 자신의 서브그룹의 키를 갱신하고 분배한다. 이 때 새로운 멤버의 OFT 트리 경로상의 sibling blinded keys는 K로 암호화해서 전달한다.

4.2.3 Leave Protocol

어떤 멤버 m이 그룹에서 탈퇴하는 경우 forward secrecy를 위해 m이 알고있는 모든 키의 정보를 갱신해야 한다. 그러므로 m이 속한 서브그룹의 SGC_j는 OFT의 그룹 탈퇴 프로토콜에 따라 키를 갱신하고 남아있는 모든 멤버들에게 안전하게 전달한다.

만일 m의 그룹 탈퇴로 인해 SGC_j의 서브그룹에 더 이상 멤버가 존재하지 않게 되는 경우 SGC_j의 역할은 없어지며 그러므로 SGC_j는 상부계층에 대한 탈퇴를 수행한다. 이때 상부계층에 대한 탈퇴는 TGDH 트리의 스폰서(sponsor) 노드가 상부계층의 키 갱신과 분배를 담당하게 된다. 필요한 경우 SGC_j는 다시 그룹의 상부계층에 참여하게 된다.



[그림 12] SGC의 상부계층 탈퇴

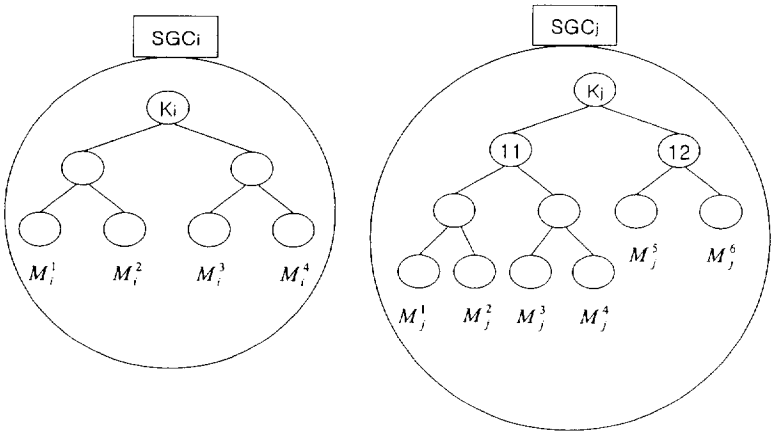
4.3 그룹 합병(Merge) 및 분할(Partition)

그룹 합병(group merge)과 그룹 분할(group partition)은 시스템 오류로 인해 어떤 SGC가 자신의 역할을 수행할 수 없거나 혹은 서브그룹의 멤버의 증가로 인해 해당 SGC에게 과다한 작업량(workload)이 부가되는 경우 SGC의 작업을 다른 *SGC*에게 분담하기 위함이다.

만일 시스템의 오류로 인해 SGC_i가 그룹을 관리할 수 없게 되는 경우 해당 서브그룹의 멤버들을 다른 서브그룹에 이전시키고 이로 인한 키의 갱신을 통해 계속적인 안전한 그룹 통신을 수행할 수 있게 해야 한다.

그림 13은 SGC_i와 SGC_j의 OFT 키 트리를 보여준다. SGC_i에 오류가 발생한 경우 멤버 M_i^*들은 SGC_j의 관할로 이동하며 해당 서브그룹의 키 트리를 유지하면서 서브그룹의 키 정보를 그대로 사용하도록 한다. 즉 그룹 합병을 위한 SGC_i의 키 트리의 깊이를 h_i , SGC_j의 키 트리의 깊이를 h_j라하고 $h_i \leq h_j$라면, SGC_j

는 자신의 키 트리에서 h_i위치의 노드의 하부트리(sub-tree)로 SGC_i의 키 트리를 추가하게 된다. 만일 $h_i \geq h_j$이면 SGC_i의 키 트리의 하부트리로 SGC_j의 키 트리를 추가한다. 이 때 노드의 분할은 OFT 방식을 따른다.



[그림 13] SG_i 와 SG_j의 키 트리 구성

그림 13에서 SG_i의 키 트리는 SG_j의 키 트리에서 노드<12>에 연결되며, SGC_j는 해당 노드를 분할(splitting)하여 왼쪽 자식노드에는 기존의 노드에 대한 서브트리를 연결하고, 오른쪽 자식노드는 새로 추가되는 SG_i의 트리를 연결한다. 그림 14는 갱신된 키 트리를 보여 준다.

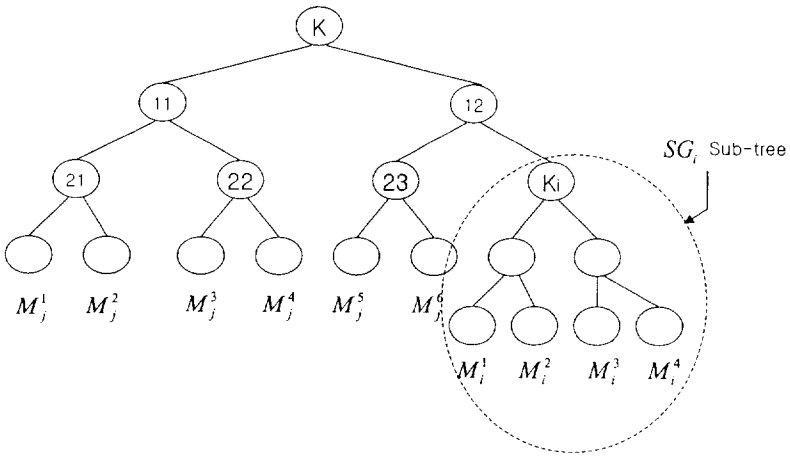


그림 4-14 그룹 합병으로 인한 키 트리 갱신

키 트리를 갱신하고 나면 SG_i 트리에서 제일 오른쪽 노드의 멤버 M_i^4는 새로운 그룹키 계산에 필요한 blinded 키들을 SGC_j에게 제공하고, SGC_i는 노드<23>의 blinded key bK_{23}과 bK_i를 이용해서 상위노드의 키를 계산하고, 계속해서 트리의 경로상의 키를 계산함으로써 서브그룹 키를 갱신한다. 그리고 bK_i로 SG_i 서브트리의 루트에서 갱신된 키 트리의 경로상의 sibling blinded key들을 암호화해서 SG_i 멤버들에게 전송한다. 그러므로 M_i^* 멤버들은 그룹키를 계산할 수 있게 된다.

SG_i의 키 트리를 SG_j의 키 트리에 h_i에 추가함으로써 가장 깊은 노드와 가장 얕은 노드의 차이가 1이므로 OFT 트리의 full-balanced tree 성질을 그대로 유지하게 된다.

/* occurs system fault in SGC_i */

M_i^*s migrate to SGC_j

SGC_j reconstruct key tree

finds attaching point at h_i(if $h_i \leq h_j$) level

splits node into left child and right child

existing sub-tree is linked left child

SG_i's key tree is linked right child

leftmost M_i^l sends SG_i's blinded keys $bSGK_i$ s to SGC_j

SGC_j compute SG_j's group key SGK_j

$SGC_j \Rightarrow M_i^*$: encrypted sibling blinded keys

$SGC_j \Rightarrow M_j^*$: encrypted updated blinded keys

[Procedure 4] 그룹 합병으로 인한 서브그룹의 키 트리 갱신

다음은 그림 14에 대한 갱신된 키의 전달 과정을 나타낸다.

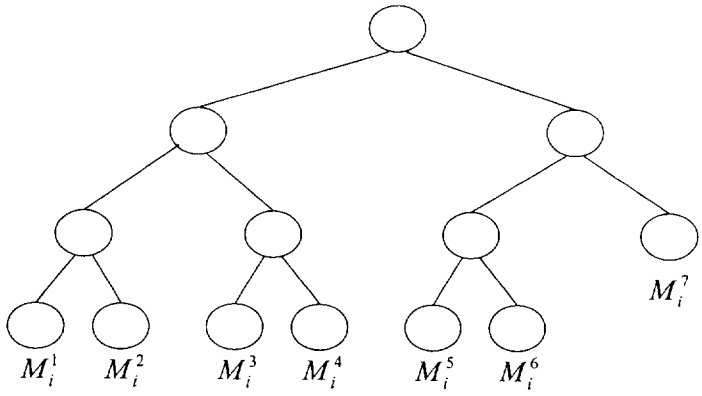
$SGC_j \Rightarrow M_i^*$: $\{bK_{23}, bK_{11}\}_{bK_i}$

$SGC_j \Rightarrow \{M_j^1, M_j^2, M_j^3, M_j^4\}$: $\{bK_{12}\}_{K_{11}}$

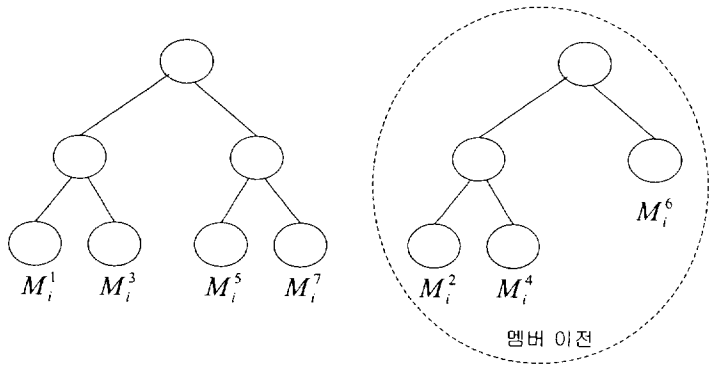
$SGC_j \Rightarrow \{M_j^5, M_j^6\}$: $\{bK_i\}_{K_{23}}$

만일 SGC_i의 멤버수의 증가로 인해 멤버들의 관리를 위한 작업량이 과다한 경우 작업량이 적은 다른 서브그룹으로 일부 멤버들을 이전시키기 위해서, SGC_i는 그룹 분할을 통해 이전되는 멤버들과 남게 되는 멤버들에 대한 OFT 키 트리를 새로 생성하고 이전되는 서브그룹의 SGC_j에게 이전되는 멤버들에 대한 키 트리의

정보와 새로운 그룹키 계산에 필요한 blinded key들을 제공함으로써 그룹을 재구성하도록 한다.



[그림 15] 서브그룹 SG_i의 그룹 분할 이전



[그림 16] 서브그룹 SG_i의 그룹 분할

그림 15와 그림 16은 서브그룹 SG_i의 그룹 분할에 대한 키 트리의 분할 과정을 나타낸다. 분할된 멤버들은 이전하는 해당 SGC_j와 그룹 합병 절차에 따라

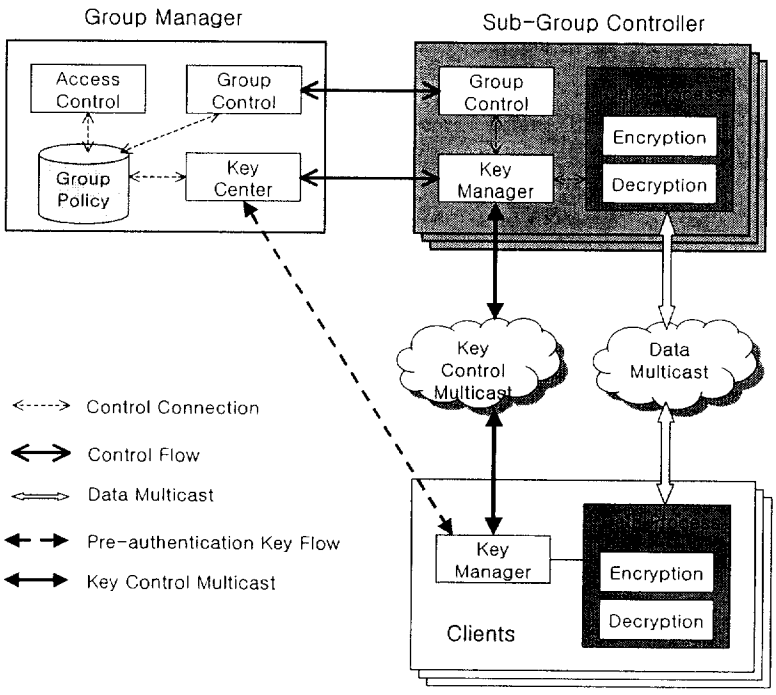
SGC_j의 서브그룹에 소속된다.

4.4 구현방안 및 고찰

본 논문에서 제안하는 2계층(2-tier)방식의 그룹키 관리를 위한 시스템 구조는 그림 17과 같다. 그룹 관리자 *GM*은 전반적인 그룹관리를 위한 정책을 설정하고 키 센터(key center)를 통해 각 서브그룹 관리자와 멤버들간의 키 인증을 위한 키 파라미터를 관리한다. 각 서브그룹 관리자 *SGC*는 group control 프로세스들간의 통신을 통해 그룹 합병 및 분할 작업을 수행한다. 각각의 서브그룹마다 별도의 멀티캐스트 IP 주소를 사용해서 통신을 수행할 수 있고, *SGC*는 자신의 그룹과 서브그룹 관리자들로 구성되는 상부계층의 관리그룹에 모두 참여함으로써 각 서브그룹간의 통신을 중재하도록 한다.

그림 17의 시스템에서 그룹 관리(Group control) 프로세스들은 전체 시스템의 작업분담(load-balancing)을 위해 통신하게 된다. 만일 어떤 서브그룹 관리자가 관리해야 하는 멤버의 수가 과다한 경우 그룹 관리 메시지의 전달을 통해 상대적으로 작업 부담이 적은 다른 서브그룹 관리자의 그룹으로 멤버들을 이전시키도록 한다.

각 계층그룹과 하부의 서브그룹들은 서로 독자적인 키 관리 기법을 사용할 수 있으며, 본 논문에서는 SGC들에 대한 상부계층의 *CGK*를 위한 그룹키 관리는 TGDH를 사용하고 서브그룹내의 *SGK*를 위한 그룹키 관리는 OFT를 사용하여 구현하였다. TGDH는 법지수 연산으로 인한 계산량이 많으나 decentralized 방식의 장점을 이용해 중앙 키 관리 서버에 대한 'one-point failure'를 예방할 수 있다.



[그림 17] 그룹 관리 시스템 구조

멤버의 그룹 참여를 위한 *GM*의 pre-authentication과정은 SSL[15]과 같은 안전한 유니캐스트 채널을 통해 수행되고, 멤버의 key agreement를 위한 ICPK를 멤버에게 제공한다. ICPK의 안전한 전달을 위해 멤버의 pre-authentication에서만 SSL이 사용되고 key agreement 단계에서는 SSL을 사용되지 않는다. 표 5는 한번의 키 설정에 대한 사용자의 계산 시간을 비교해서 보여준다. 실험을 멤버측의 구현은 Pentium II 500MHz 시스템에서 Java를 사용하여 구현되었으며, SSL은 Sun의 JSSE1.0.2(Java Secure Socket Extension 1.0.2)를 사용하여 구현하였다. 전체적인 계산을 비교해 볼 때, AGKA와 SSL 채널을 이용한 계산에 큰 차이가 없으나 AGKA는 단지 초기 인증과정에서만 SSL 채널을 이용하며 이 후의 계속적인 키 설정과정에서는 SSL의 사용에 대한 오버헤드가 제거된다.

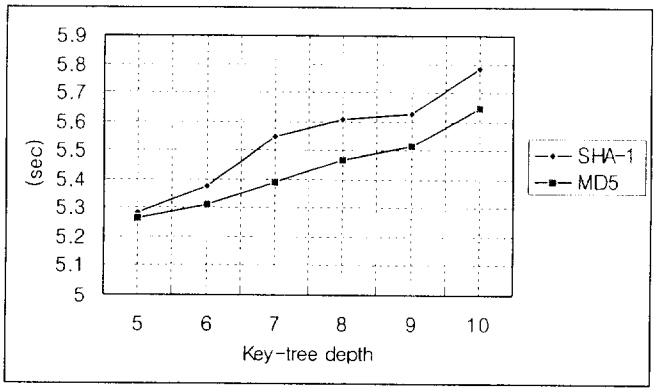
[표 5] 키 설정 시간 비교

키 설정 방식	AGKA 이용		Secure Channel 이용
	pre-authentication(SSL)	Key-agreement	Key-agreement(SSL)
시간(s)	7.138	0.981	8.372

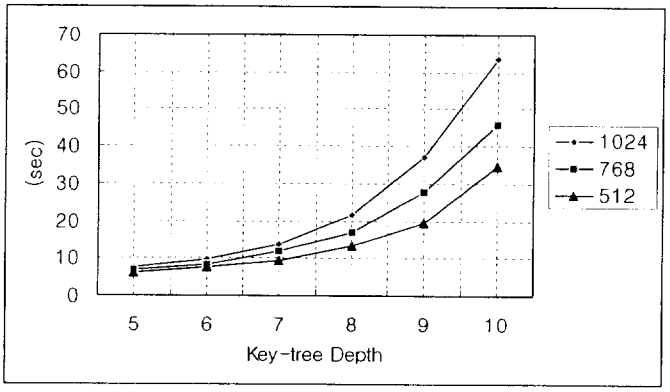
키 파라미터 생성을 위한 소수 p의 크기는 일반적으로 1024비트 이상이면 안전하다고 알려져 있으나, 지수연산은 많은 계산량을 요구하므로 그룹키 생성을 위한 지수의 크기가 1024비트 크기일 필요는 없으며 법-지수(modular-exponentiation) 연산으로 인한 1024비트 크기의 키를 암호학적 해쉬함수를 적용하여 실제 그룹키 계산을 위해 128비트 혹은 160비트 키를 법지수 연산의 지수로 사용함으로써 계산량을 감소시킬 수 있다. 그러므로 TGDH 계산을 위한 실제 계산과정은 다음과 같이 수행된다.

$$
\begin{aligned}
K_{(l,v)} &= (\alpha^{h(K_{\langle l+1,v_l\rangle})})^{h(K_{\langle l+1,v_R\rangle})} \bmod p \\
&= (\alpha^{h(K_{\langle l+1,v_R\rangle})})^{h(K_{\langle l+1,v_l\rangle})} \bmod p \qquad ; \text{h() : hash function} \\
&= \alpha^{h(K_{\langle l+1,v_l\rangle}) \cdot h(K_{\langle l+1,v_R\rangle})} \bmod p
\end{aligned}
$$

그림 18과 그림 19는 서버측에서 OFT와 TGDH를 이용해서 키 트리를 구성하고 그룹키를 계산하는데 소요되는 시간을 보여준다. 각각의 실험은 Pentium III 1GHz 시스템에서 수행되었으며, OFT는 일방향함수로 암호학적 해쉬함수인 MD5와 SHA-1을 사용하고 혼합함수로 bit-XOR을 사용했을 때의 계산 시간을 나타내고 있고, TGDH 계산은 법(modular) p를 각각 1024, 768, 512비트의 소수로 사용하고 지수는 키에 MD5 해쉬함수를 적용하여 128비트 키를 지수로 했을 때의 계산 시간을 보여준다.

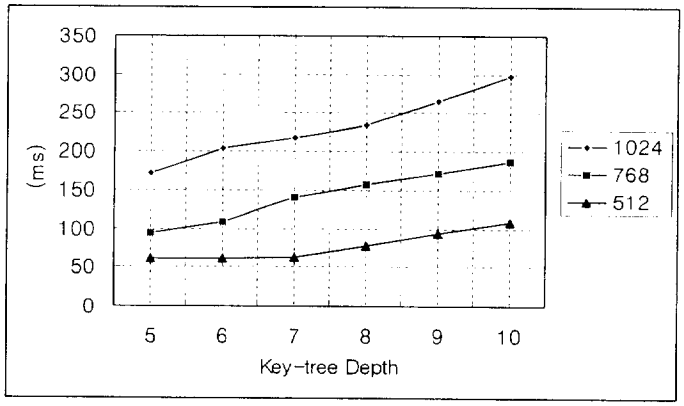


[그림 18] OFT 키 트리 계산 시간

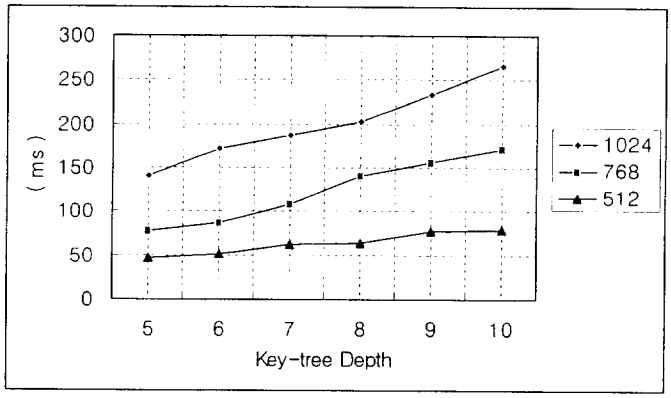


[그림 19] AGKA/TGDH 키 트리 계산 시간

AGKA와 TGDH는 지수연산으로 인한 계산이 많이 소모되나 AGKA와 TGDH가 상부계층의 그룹키 관리를 위한 메커니즘임을 고려할 때, 전체 그룹에서 각 서브그룹 관리자의 수는 많지 않을 것이고 서브그룹 관리자는 그룹 관리를 위한 충분한 컴퓨팅 능력을 가진 서버급 시스템에 의해 계산될 것이므로 전체 계산이 많이 소요되지는 않을 것이다. 그리고 TGDH의 키 관리는 분산형방식(decentralized)이므로 어느 서브그룹 관리자나 키 분배에 대한 기능을 수행할 수 있는 특징을 가지게 된다.



[그림 20] 멤버 추가시 TGDH 갱신 시간



[그림 21] 멤버 삭제시 TGDH 갱신 시간

그림 20과 그림 21은 TGDH에서 멤버의 추가와 탈퇴에 대한 키 트리 갱신에 소요되는 시간을 나타낸다.

OFT 키 트리에서 멤버의 추가와 탈퇴로 인한 키 트리의 갱신은 평균 5ms이하의 시간이 소요되었으며, TGDH의 경우 법지수 연산으로 인한 멤버의 추가나 삭제시의 계산이 OFT보다 많이 소요된다. 표 6은 OFT와 TGDH에서 키 갱신에 대한 계산량을 보여준다.

[표 6] 키 갱신 계산량

	TGDH	OFT
키 갱신 메시지 크기	$(\log_2 n) \cdot \lvert K_T \rvert$	$(\log_2 n) \cdot \lvert K_O \rvert$
키 갱신 연산 횟수	$O(\log n)$	$O(\log n)$

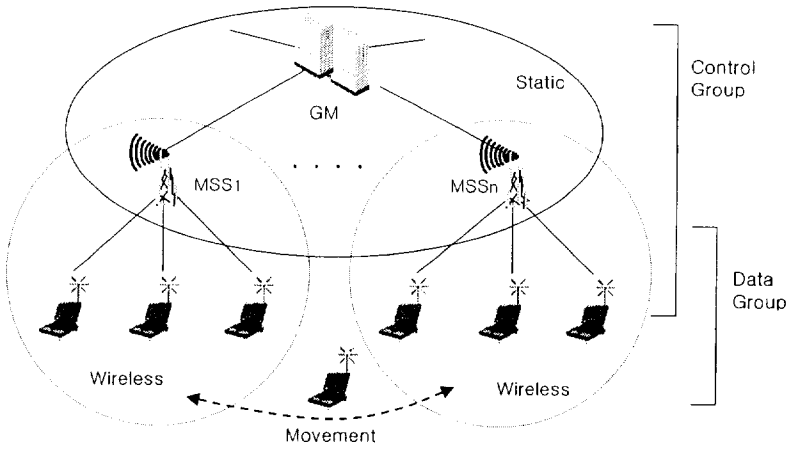
$\lvert K_T \rvert$: TGDH 노트키 크기 $\lvert K_O \rvert$: OFT 노드키 크기

4.5 활용방안

4.5.1 이동네트워크 환경에서의 그룹키 관리구조

이동네트워크 환경에서의 그룹의 구성은 전체 그룹의 관리를 위한 그룹 관리자 *GM*과 이동호스트로 구성되는 그룹 멤버 그리고 이동호스트가 속한 셀(Cell)에서 이동호스트로의 데이터 전송을 담당하는 *MSS*(Mobile Support Station)들로 구성된다[16]. 이동네트워크 환경에 2-tier 방식의 그룹 구조를 적용할 때, 첫 번째 계층은 *GM*과 각 셀의 MSS_i들로 구성되는 제어 그룹이 되며 *GM*과 MSS_i는 유선네트워크를 통해 통신하며, 두 번째 계층은 각 *MSS*와 이동호스트들로 구성되는 셀 수준의 데이터 그룹으로 볼 수 있다. 그림 22는 이러한 이동네트워크에서의 그룹 구조를 나타낸다.

그림에서 각 셀의 MSS_i는 해당 셀내의 이동호스트들로 구성되는 데이터 그룹을 관리하며, *GM*은 전반적인 그룹 관리와 각 셀의 MSS_i들로 구성되는 제어그룹에 대한 관리를 담당한다. MSS_i는 자신의 셀내에 그룹 멤버로 등록된 호스트가 존재하는 경우에만 상위계층의 제어 그룹의 구성원으로 참여하게 되며 멤버 호스트가 존재하지 않은 경우 제어그룹에서 탈퇴하게 된다.

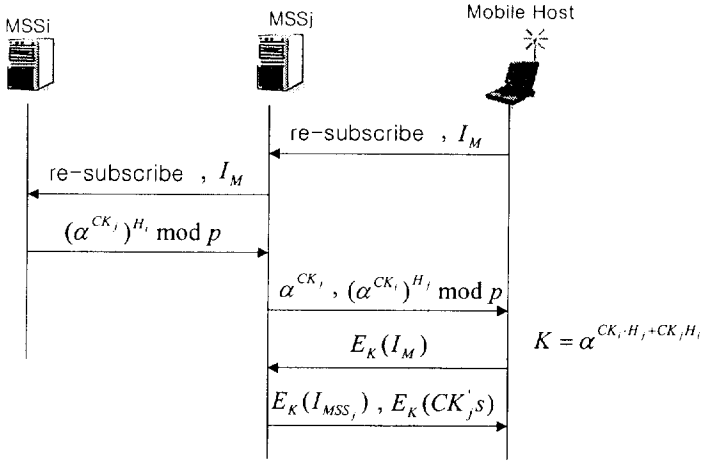


GM : Group Manager MSS : Mobile Support Station MH : Mobile Host

[그림 22] 이동네트워크에서의 그룹 구조

이동네트워크의 주된 특징은 호스트의 이동성(mobility)이며, 어느 셀에서 다른 셀로의 호스트의 이동은 셀 수준(Cell level)에서, 이전 셀에 대해서는 해당 셀 그룹의 탈퇴이며 새로 이동한 셀에 대해서는 그룹 참여가 수행되어야 한다. 그러므로 호스트의 이동성은 그룹의 동적인 특성을 더욱 복잡하게 하며, 호스트의 이동성에 대해서도 확장성을 제공하는 그룹관리가 이루어 져야 한다.

현재 MSS_i의 영역에 속한 이동호스트 MH가 MSS_j의 영역으로 이동하는 경우 MH는 MSS_j를 통해 그룹에 재참여(re-subscribe)하게 되고, MSS_i와 MSS_j간에 hand-over가 발생하게 되며 이동호스트에 대한 키 설정 프로토콜은 MSS_i와 MSS_j 그리고 MH간에 이루어진다. 그림 23은 hand-over에서의 키 설정 프로토콜을 나타낸다.



CK : Cell-group Key

$H_i = (CK_i, I_{MSSi} \| I_{MSSj} \| I_{MH_i})$

$H_j = (CK_j, I_{MSSi} \| I_{MSSj} \| I_{MH_i})$

[그림 23] Hand-over 과정에서의 키 설정

키 설정을 위한 파라미터인 소수 p와 Z_p^*의 생성자 α는 GM이 생성하고 공개하며, 각 MSS_i는 자신과 셀내의 멤버들간에 공유되는 셀 그룹키 CK_i에 대한 α^{CK_i}를 공개하도록 한다. 호스트가 이동하는 경우 hand-over를 통해 새로운 키를 설정하게 되며, H_i는 이전 셀 그룹의 CK_i를 알고있는 멤버만이 계산할 수 있으므로 MH에 대한 묵시적인 인증이 수행될 수 있다. 이동호스트 MH는 키를 계산하기 위해 한번의 암호학적 해쉬연산과 법-지수연산을 수행하게 된다.

멤버 호스트 MH가 MSS_j의 영역에서 그룹의 재가입 요청은 MH의 전체 그룹에 대한 멤버십은 변경되지 않고 자신의 관리 영역만이 변경되므로 GM에 대한 pre-authentication은 수행되지 않아도 된다.

이전 셀 그룹의 MSS_i는 MH의 이동에 대해 셀 그룹의 탈퇴로 인한 셀 그룹키를 갱신하고 남아있는 멤버들에게 안전하게 전달하며, 새로 진입한 셀 그룹의

MSS_j는 설정된 키를 MH의 비밀키로 하여 셀 그룹키를 갱신하고 멤버들에게 갱신된 키를 전달한다. 만일 MH의 이동으로 인해 이전 셀 그룹 MSS_i의 영역에 더 이상 그룹 멤버가 존재하지 않는다면 MSS_i는 제어그룹에서 탈퇴하게 되고, 새로 진입한 셀 그룹의 MSS_j에 기존의 멤버가 없었다면 MSS_j는 먼저 제어그룹에 참여하고 MH와 키를 설정하고 셀 그룹을 구성한다. 그러므로 GM은 제어그룹의 참여와 탈퇴에 대한 상위계층의 키를 갱신하고 분배해야 한다.

5. 결론

현대 컴퓨터 네트워크 기술의 발전으로 인해 사용자들의 인터넷을 통한 다양한 서비스에 대한 요구와 이러한 요구를 충족시키기 위한 여러 어플리케이션이 등장하고 있다. 이로 인해 동일한 서비스를 여러 사용자들에게 동시에 제공해 주기 위한 그룹 통신을 위한 어플리케이션들이 등장하고 있으며 멀티캐스트는 이러한 그룹 통신에 적합한 프로토콜이다. 그러므로 안전한 멀티캐스트 통신을 위한 그룹키의 관리가 중요하며, 그룹의 관리에 있어서 안전성, 효율성 그리고 확장성 등을 만족해야 한다.

본 논문에서는 안전한 그룹통신을 위해 필요한 요구사항들과 그룹키 관리기법들을 고찰하고, 다양한 그룹키 관리 기법들을 토대로 그룹의 특성에 따라 효율성과 확장성을 제공하기 위한 그룹키 관리 구조를 설계하였다. 본 논문에서 제안하는 그룹 관리의 계층화는 전반적인 그룹 관리를 각각의 서브그룹 관리자들에게 분담하고 각 서브그룹을 독립적으로 관리하며, 한 서브그룹에서의 멤버십의 변경으로 인한 영향을 지역적으로 한정시킴으로써 다른 서브그룹에 영향을 주지 않도록 확장성(Scalability)를 제공한다.

초기인증 이후의 authenticated key agreement를 위해 ICPK를 사용함으로써 인증서의 교환과 검증이 요구되지 않으므로 효율적인 key agreement를 수행할 수 있다. 그리고 그룹 분할과 합병을 통해 각 서브그룹 관리자의 작업을 공평하게 분담할 수 있으며, "one-point failure problem"을 방지할 수 있는 특성을 가진다.

본 논문에서 제안된 그룹키 관리구조가 이동네트워크 환경에서의 안전한 그룹 통신을 위한 그룹 관리 형태로 적용될 수 있을 것으로 기대하며, 이동 호스트의 빈번한 이동에 대한 효율적인 그룹키 관리에 대한 연구도 지속적으로 수행되어야 할 것이다.

참 고 문 헌

[1] T. Hardjono, G. Tsudik, IP Multicast Security: Issues and Direction, Annales de Telecom, 2000

[2] S. Mittra, Iolus: A Framework for Scalable Secure Multicasting, Proceedings ACM SIGCOM, p.277-288, 1997

[3] T. Hardjono, Cain, B., N. Doraswamy, A Framework for Group Key Management for Multicast Security, IETF Internet draft, Feb, 1999

[4] D. Balenson, D. McGrew, and A. Sherman, Key management for large dynamic groups: One-way function trees and amortized initialization. Internet Draft: draft-balensongroupkeymgmt -oft-00.txt, February 1999

[5] M. Waldvogel, G. Caronni, D. Sun, N. Weiler, and B. Plattner, The VersaKey Framework: Versatile Group Key Management. Selected Areas in Communications, IEEE Journal on , Volume: 17 Issue: 9, p.1614-1631, Sept. 1999

[6] C. K. Wong, M. Gouda, and S. S. Lam, Secure group communications using key graphs. In Proc. ACM SIGCOMM, p.68-79, Aug. 1998

[7] C. K. Wong, S. S. Lam, Digital signatures for flows and multicasts, IEEE Transactions on Networking, Volume 7 Issue: 4, p.502-513, Aug, 1999

[8] L. R. Dondeti, S. Mukherjee, and A. Samal, A dual encryption protocol for scalable secure multicasting. Proceedings of the 4th International Symposium on Computer and Communications, p.2-8, Jul. 1999

[9] Y. Kim, A.Perrig, G. Tsudik, Simple and Fault-Tolerant Key Agreement for Dynamic Collaborative Groups, Proceedings of the 7th ACM conference on Computer and communications security, p.235-244, Nov. 2000.

[10] 박희운, 이임영, 효율적인 멀티캐스트 키 관리 구조 제안 및 효율성 분석, 2001년 한국정보처리학회 춘계학술발표논문집 제8권 제1호 p.367-370, 2001년 4월

[11] H. Harney, E. Harder, Logical key hierarchy protocol, Internet Draft, IETF, April 1999

[12] A. Perrig, Efficient collaborative key management protocols for secure autonomous group communication. International Workshop on Cryptographic Techniques and E-Commerce CrypTEC '99.

[13] C. G. Gunther, An identity-based Key Exchange Protocol. In EUROCRYPT89, 1989

[14] Hyun-Ho Cho, Young-Ho Park, Jun-Seok Lee, Hwa-Sik Jang, Kyung-Hyune Rhee, "A Proposal of Secure Efficient Dynamic Hierarchical Key Management Structure", The 2001 Workshop on Information Security Applications (WISA2001), p.357-362, Sep. 2001.

[15] A. O. Freier, P. Karlton, P. C. Kocher, The SSL Protocol Version 3.0. Netscape Communications, Nov. 1996

[16] D. Bruschi, E. Rosti, Secure multicast in wireless networks of mobile hosts: protocols and issues, ACM-Balzer MONET Journal, Special issue on Multipoint Communication in Wireless Mobile Networks, 2000

[17] Y. Zheng, Shortened Digital Signature, Signcryption and Compact and Unforgeable Key Agreement Schemes, Submission to IEEE P1363a: Standard Specifications for Public-Key Cryptography, August 1998.

감사의 글

'감사의 글'. 동네 사람들!, 드디어 호야한테도 이 글을 적게 되는 날이 왔습니다. 지난 2년 동안 선배들의 감사의 글을 읽으며 기다려왔던 날입니다. 공부에 대한 의욕이 반이요, 세상에 나아가는 데 대한 두려움이 반으로 시작되었던 석사 생활이 어느덧 2년이라는 시간을 뒤로하고 있습니다.

무엇보다도 항상 제가 하고자 하는 것을 믿고 묵묵히 지켜봐 주신 부모님께 감사드립니다. 공부라는 핑계로 가계에 도움이 되지 못하고 다시 진학을 선택했을 때에도 아무말씀 없이 제 뜻을 따라 주신데 대해 또 감사 드리며, 한편으로는 죄송한 마음도 생깁니다.

저에게 새로운 학문의 길을 열어주시고 저를 이날까지 이끌어주신 이경현 교수님께 감사 드립니다. 그리고 제 논문을 심사해주신 김창수 교수님, 김영봉 교수님, 윤성대 교수님과 제 학업을 지도해주신 박만곤 교수님, 박지환 교수님, 윤성대 교수님, 박홍복 교수님, 박승섭 교수님, 여정모 교수님, 정순호 교수님께 감사 드립니다.

석사생활동안 나의 추상적 이상향이었던 원이 선배 올해에는 장가가시길..... 우리 연구실의 행동대장에서 실땅님이 되신 준석이 선배와 잘 개기던 이 후배와 이제는 동기가 될(?) 종필이 선배 고맙습니다. '샘~'에서 '선배'가 되어버린 정화 선배, 막강 파워의 아줌마! 양수정 샘 감사합니다. 남들은 못 믿겠다는 나의 동기 현호야, 고맙다. 잘 나지도 않은 나를 선배로 만나 고생한 석사 후배 지철이, 명진이, 희연이, 이번에 새식구가 된 성렬이와 영경이, 학부생이지만 나보다 연구실 짬밥(?)이 높은 미선이, 뿌~웅 오리 정훈이, 가끔씩 지치거나 힘들고 짜증날 때 옆자리에서 나를 웃음짓게 했던 종금이, 모두들 고마우이. 그리고 함께 공부해왔던 연구실의 산대원, 교대원 선생님들께도 감사의 말씀을 드립니다.

마지막으로 우리 93학번 동기들. 새신랑 년조, 영재, 성환이, 용섭이 이제 5월의 신부를 꿈꾸는 행진이와 은영이 그리고 철이, 정우, 수범, 현호, 태호. "친구들아 고

맙데이. 너거들이 함께 있어서 두려울게 없다."

언제가 신원 선배의 박사학위 논문의 감사의 글이 '감사의 글 2'였던 것이 기억납니다. 원이 선배의 석사학위 논문의 감사의 글이 1편이고 박사학위의 감사의 글이 2편이었다더군요. 이제 저도 제 감사의 글에 대한 속편을 위한 그 날을 위해 새로이 시작하려고 합니다.

To be continue . . . . . .

2002년 새해가 뜬지 며칠 되지 않은 1월의 어느 새벽에.....