한국산학기술학회논문지
Vol. 12, No. 1 pp. 436-444, 2011

멀티캐스트 전송을 위한 에이전트 기반의 안전한 그룹 키 관리방안 연구

김보승[1*], 김정재[1], 장봉덕[2], 신용태[1]
[1]숭실대학교 컴퓨터학과, [2]에이치텔레콤(주)

A Study on Secure Group Key Management Based on Agent for Multicast Data Transmission

Bo-Seung Kim[1*], Jung-Jae Kim[1], Feng-De Zhang[2] and Yong-Tae Shin[1]
[1]Dept. of Computing, Soongsil Univ., [2]Htel Inc.,

요 약 최근 멀티캐스팅의 활용도가 높아짐에 따라 그에 대한 보안이 중요한 문제로 인식되게 되었다. 멀티캐스트 환경에서 보안성을 제공하기 위해 지금까지 진행되어 오고 있는 연구는 주로 그룹 키 관리기법에 관한 것이다. 멀티캐스트 보안에 있어서 가장 중요한 것은 허가된 멤버만이 데이터에 접근할 수 있어야 한다. 이는 적합한 그룹 멤버들만이 공유하는 그룹 키를 통해 데이터를 암호화하여 전송함으로써 해결된다. 본 논문에서는 안전한 멀티캐스트 데이터를 전달하기 위하여 그룹 가입/탈퇴가 빈번한 멀티캐스트 환경에 적합한 에이전트기반의 그룹 키 관리방안을 제안하고자 한다. 에이전트기술을 도입하여 전체그룹을 여러 개의 서브그룹으로 분할하여 에이전트가 각 서브그룹의 멤버들을 관리하도록 한다. 또한 에이전트 내부에서는 트리기반으로 키를 관리하고 키 갱신을 위하여 일방향 해쉬함수를 사용한다. 이러한 제안방식은 통신 중간단계에서의 그룹 키 교환을 배제하여 키 노출을 방지하고 키 재분배 과정에서 발생하는 지연을 감소시킬 수 있다.

Abstract As practical use degree of multicasting increase, security for multicast is recognized as an important issue. Previous research in the area of secure multicast has mainly focused on group key management. The most important thing about the security of multicast is that only authorized members of this group will be able to access the data. The member of access to multicast communication is to use cryptography with a common shared session encryption key. We propose decentralized group key management based on agent for dynamic multicast with large groups and frequent joins or leaves in this paper. Whole group divide to several subgroup using agent technology and each agent manage members of each subgroup. Also, when rekeying updates that using one-way hash function can prevent the key exposure, and reduce the key distribution delay.

Key Words : Group-Key, Multicast, Security, Agent

1. 서론

인터넷의 급속한 발전에 따라 IPTV, 화상회의, 인터넷 방송 등 그룹통신에 기반 한 새로운 종류의 통신 응용 서비스들이 날로 증가하고 있다. 이러한 그룹통신은 일대다 혹은 다대다 통신 형태로 구성되며 멀티캐스트는 이러한 그룹통신에 적합한 프로토콜이다. 멀티캐스트 환경에서는 메시지에 대한 하나의 복사본만을 전송함으로써 멀티캐스트 그룹의 모든 멤버가 수신할 수 있고 단 하나의 그룹 키로 멀티캐스트 통신을 보호받을 수 있다. 멀티캐스트 그룹 키는 멤버들의 그룹 가입/탈퇴로 인하여 그룹 키 생성 및 재분배가 필요하다. 또 안전한 그룹 키 관리를 위해서는 주기적으로 키 갱신이 이루어져야 한다. 그러나 그룹의 규모가 큰 경우 모든 멤버에게 그룹 키를 재분배해야 하기에 네트워크 지연을 발생시키는 단점을 가진다.

본 논문은 안전한 멀티캐스트 데이터 전송을 위하여

*교신저자 : 김보승(kdwon2002@ssu.ac.kr)
접수일 10년 10월 25일 수정일 (1차 10년 12월 03일, 2차 11년 01월 01일) 게재확정일 11년 01월 13일

그룹 가입과 탈퇴가 빈번한 멀티캐스트 환경에서 에이전트기반의 그룹 키 관리기법을 제안한다. 멀티캐스트 프로토콜의 시작은 IGMPv3[1]을 통하여 이루어지며 세션 세업 시 첫 번째 그룹 키의 생성은 AES 알고리즘을 통하여 완성이 된다. 상대적으로 부하가 많은 공개키 알고리즘은 사용자의 정보를 확인하기 위해 사용되며, 새로운 사용자가 추가되거나 탈퇴 시 난수값(Nonce) 전송을 위해 AES 알고리즘을 사용한다. 이때 난수값을 암호화 과정없이 전송할 때 불법사용자가 메시지를 가로채어 사용하는 것을 막기 위해 암호화 알고리즘을 사용한다. 또한 에이전트기술을 도입하여 서버를 분산시켜 멤버들을 관리하도록 하며, 에이전트의 내부구조에 트리기반의 키 관리기법을 제시하고 위에 언급되었던 Signed token을 추가하는 것으로 기존 키 관리기법에서 멤버인증의 단점을 해결한다. 또한 키 갱신과정에서 해쉬함수를 이용함으로써 통신 중간단계의 키 노출을 방지하고 키 재분배 과정에서 발생하는 지연을 감소시키는 관리기법을 제안한다.

본 논문의 구성은 다음과 같다. 2장은 관련연구로 멀티캐스트 보안과 기존 그룹 키 관리기법들을 분석한다. 3장에서는 새로운 키 관리방안을 제안하는 멀티캐스트 환경에서의 에이전트기반 그룹 키 관리에 대하여 기술한다. 4장은 수식을 통해 기존의 기법과 제안하는 기법을 비교분석한다. 마지막으로 5장에서는 논문의 결론을 맺고 향후 연구 방향을 살펴본다.

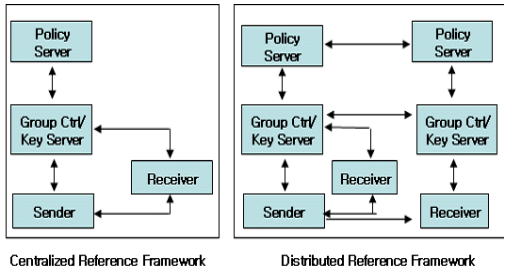
2. 관련 연구

2.1 멀티캐스트 보안

멀티캐스트가 효율적인 그룹통신을 제공하지만 보안과 관련하여 고려되어야 할 사항은 다음과 같다.

첫째, 데이터 기밀성을 제공해야 한다. 기밀성은 안전한 멀티캐스트 통신의 가장 기본적인 사항으로서 그룹 멤버간의 암호화에 사용되는 키를 공유함으로써 제공될 수 있다. 둘째, 그룹키의 유효시간을 정의하여 주기적인 키 갱신을 수행해야 하며 멤버 가입과 탈퇴 시 Backward Secrecy와 Forward Secrecy를 보장하기 위해 키 갱신이 이루어져야 한다. 셋째, 허가된 멤버들만이 그룹에 접근 가능하도록 그룹가입 시 인증이 필요하다. 넷째, 키 관리자와 멤버사이에 유니캐스트 채널을 통한 통신 이외의 방법으로 그룹 키의 재분배가 가능해야 한다. 마지막으로 데이터가 전송 도중에 변조되는 것을 막기 위해 데이터에 대한 무결성도 제공해야 한다[2].

안전한 멀티캐스트 통신을 위한 그룹 키 관리구조는 주로 두 가지가 있다. 즉 중앙집중형 키 관리구조와 분산형 키 관리구조이다. 중앙집중형은 말 그대로 GCKS(Group Controller/Key Server)와 정책 서버(Policy Server)를 각각 하나만 두어 그룹 멤버를 관리하는 구조이며 분산형은 GCKS와 정책 서버를 분산시켜 여러 개로 나누어 그룹 멤버를 관리하는 구조로서 네트워크 부하를 줄인다. 아래 그림 1은 중앙집중형과 분산형 키 관리구조를 나타내고 있다. 여기서 GCKS 키 관리 및 사용자 인증기능을 수행한다. 정책 서버는 엔티티가 사용할 보안 정책을 생성하고 관리한다. Sender/Receiver는 사용자와 장치간의 인증 및 키 생성에 필요한 값을 통하여 새로운 키를 획득한다.



[그림 1] 중앙집중형과 분산형 키 관리구조

2.2 기존의 키 관리 기법

중앙집중형은 어떤 하나의 신뢰되는 개체가 모든 그룹의 멤버들을 관리하는 것으로서 그룹의 멤버십이 변경될 때마다 그룹 관리자가 갱신된 그룹 키를 모든 그룹 멤버들에게 안전하게 전달하는 방식이다. 대표적인 기법으로 LKH[3], OFT[7] 등이 있다. 분산형은 전체 그룹을 여러 개의 서브그룹으로 나누어 멤버들을 관리하는 방식으로 멤버십이 변경될 때 해당 서브그룹의 관리자만 키 갱신을 하며 전체 그룹에 영향을 미치지 않는다. 대표적인 기법으로는 Iolus 기법[4] 등이 있다.

LKH(Logical Key Hierarchy) 기법은 중앙서버가 논리적인 키 트리구조를 유지하며 모든 키를 관리한다. 어떤 멤버가 그룹에 새로 참여하는 경우 그룹 관리자는 그에 연관되는 노드를 생성하고 그 노드에 임의로 선택한 키를 가입된 멤버에게 전달한다. 그리고 새로운 멤버로 인한 변경되는 키들을 기존의 멤버에게 전달한다. 어떤 멤버가 그룹을 탈퇴할 경우 그 노드의 형제노드가 부모노드로 대체된다. 다음 서버는 형제노드에게 새로운 키를 할당하고 경로상의 모든 키를 갱신하여 기존 사용자들에게 전달한다. LKH 기법의 문제점은 첫째, 멤버의 수가 많을 때 키 관리 서버의 부하가 크며 “one point failure” 문제가 쉽게 발생한다. 둘째, 그룹 멤버의 가입과 탈퇴로

인하여 보다 많은 양의 키 갱신 메시지를 멤버들에게 멀티캐스트 해야 한다는 것이다.

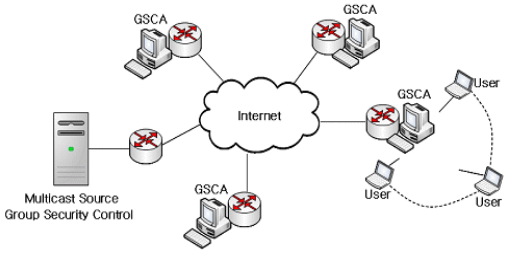
Iolus 기법에서는 그룹을 네트워크상의 노드간의 분산형 계층구조로 분할함으로써 키 관리의 효율성을 제공한다. 계층적인 키 분배를 위한 보안 관리자로써 GSC(Group Security Controller)와 GSI(Group Security Intermediary)를 이용한다. 새로운 멤버의 가입과 탈퇴 처리 작업은 그룹 전체가 아닌 해당 서브지역에서만 그룹 키를 재분배함으로써 향상된 키 관리를 제공하고 있다. Iolus에서 멀티캐스트 그룹의 초기화는 GSC와 GSI에 의해서 실행되고 GSI는 GSC의 주소를 알아야 한다. 처음에는 어떠한 GSI도 실제적으로 멀티캐스트 그룹에 가입하지 않는다. 인증된 집단이 그룹에 가입을 원한다는 것을 GSI에 알리면 GSI는 새로운 서브 그룹 키를 생성하고 이를 보호 유니캐스트 채널을 통해 새 멤버에게 전달한다. 그 후에 GSI는 계층구조에서 다음으로 높은 서브그룹에 가입한다. Iolus 기법의 문제점은 첫째, 멀티캐스트 그룹이 커질수록 많은 GSI들을 필요로 한다. 둘째, 데이터를 전송하기 위해서 매 레벨마다 GSI에서 암복호화를 수행한다. 따라서 보호 분산트리의 레벨이 많아지면 데이터를 전송하기 위한 암복호화 횟수가 늘어나 전송시간이 많이 걸린다.

3. 에이전트 기반의 그룹키 관리기법

3.1 키 관리기법의 구조

멀티캐스트 통신에서 인증, 기밀성, 무결성 등 보안 서비스를 제공하기 위해서는 안전한 그룹 키 관리구조가 필요하다. 그룹 키 관리 프로토콜의 설계는 구체적인 인터넷의 토폴로지에 따라 결정된다.

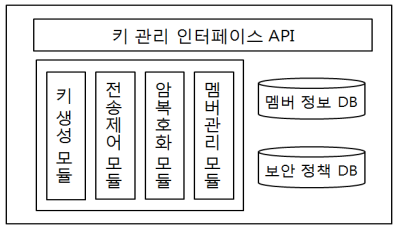
본 논문에서는 실제 네트워크 특성을 고려하여 Iolus 기법의 분산계층트리구조와 IGMPv3을 참조하여 에이전트기반의 그룹 키 관리구조를 설계한다. 그림 2는 에이전트기반의 그룹 키 관리구조를 보여준다.



[그림 2] 에이전트 기반의 그룹 키 관리구조

하나의 멀티캐스트 그룹을 여러 개의 서브그룹으로 분산시켰으며 각 서브그룹의 보안제어와 키 관리는 에이전트가 담당한다. 본 논문에서 이런 각각의 에이전트를 GSCA(Group Security Control Agent)라고 한다. 프로토콜의 구조는 하나의 GSC(Group Security Control)와 여러 개의 네트워크상에 분포되어 있는 GSCA 및 멀티캐스트 멤버들로 이루어진다. GSC와 GSCA는 네트워크상에서 분산되어 있으나 논리적으로 계층구조를 이룬다. GSC는 그룹 키 GK(Group Key)를 갖고 있고 GSCA는 GK와 서브 그룹 키 DK(Downsteam Key)를 갖고 있다. 각 GSCA는 GSC로부터 암호화된 메시지를 받아 GK를 이용하여 복호화하고 다시 DK로 메시지를 암호화해서 서브그룹내의 모든 멤버에게 전송한다.

GSCA는 아래와 같은 기능을 갖고 있어야 한다. 첫째, 데이터를 멀티캐스트 및 유니캐스트 방식으로 전송할 능력을 갖고 있어야 한다. 둘째, 새로운 가입한 멤버에 대한 인증을 할 수 있어야 하고 멤버의 가입과 탈퇴를 관리할 수 있어야 한다. 셋째, 키 성성 및 암복호화 기능을 갖고 있어야 한다. 넷째, 서브그룹의 키 트리를 관리하고, 멤버 가입, 탈퇴 시 키 갱신을 할 수 있어야 한다. 그림 3은 GSCA의 내부 구조를 보여주고 있다.



[그림 3] GSCA의 내부 구조

키 생성 모듈은 기존의 AES 암호화 알고리즘을 사용하여 첫 서브그룹 키를 생성하고 해쉬함수를 통한 키 갱신을 진행한다. 전송제어 모듈은 데이터 및 키 갱신메시지를 전송한다. 암복호화 모듈은 대응되는 키를 사용하여 데이터에 대한 암복호화를 진행한다. 멤버관리 모듈은 멤버인증 및 등록을 완성하며 멤버의 ID, 공개키 등 정보를 기록한다. 멤버 정보 DB에는 하나의 멤버참여 리스트(Participant_List)를 가지며 멤버 ID, 공개키 등 기본정보를 저장한다. 보안 정책 DB에는 키 트리 정보를 저장하며 멤버 변화에 따른 키 정보를 유지한다.

3.2 그룹 키 생성 및 분배

GSC가 네트워크상으로 멀티캐스트 그룹주소가 들어있는 GROUP_CONSTRUCT 메시지를 보낸다. 네트워크

상의 멤버들은 이 메시지를 받고 GSC에게 멤버응답메시지(MEMBER_REPORT)를 보내고 멀티캐스트 주소를 기록한다. GSC는 멤버응답메시지를 받은 후 각 멤버의 기본정보(주소, ID, 공개키 등)을 기록하고 멤버참여 리스트를 만든다. GSC는 멤버의 주소 및 분포특성에 따라 멀티캐스트 네트워크상의 몇몇 멤버들을 뽑아서 GSCA로 한다. 여기서 조건이 허락되면 선출한 GSCA는 멤버 중에서 계산능력이 강하고 강력한 안정성을 갖춘 것을 선택한다. 멀티캐스트 통신 환경이 이루어지면 GSC와 각 GSCA는 AES 알고리즘을 사용하여 각각 GK, DK를 생성한다. GSCA는 GSC가 생성한 그룹 키를 받아야만 다음 단계의 안전한 에이전트 기능을 수행할 수 있다. 또한 GSCA는 자신의 서브그룹 키를 안전하게 서브그룹내의 모든 멤버에게 전송해야 한다. GSC와 각 GSCA 및 각 멤버들은 모두 한 쌍의 공개키와 비밀 키를 갖고 있다. GSC는 각 GSCA의 공개키를 갖고 있으며 GSCA는 자신이 관리하는 멤버의 공개키를 데이터베이스에 저장하고 있다. 이런 방법의 공개키를 본 논문에서는 그룹 암호키로 사용한다. 또한 그림 4와 같이 송수신자간의 인증을 강화하기 위하여 Signed token을 이용한다. 이 Signed token으로 수신자는 메시지의 근원지와 송신자의 신분을 확인한다.

ID_{Sender}	Nonce

[그림 4] Signed Token

IDsender는 송신자 신분을 표시하는 값이고 Nonce는 난수값을 이용하여 항상 새로운 메시지가 발생할 수 있도록 하기 위한 값이다. Signed token은 전송할 메시지 내에 포함되며 송신자의 개인키로 전자서명 다. 아래 {M}k 기호를 도입하는데 이것은 키 k로 내용 M에 대하여 암호화한다는 의미이다. GSC와 GSCA간의 그룹 키 분배를 예로 설명한다. GSC는 GSCA에게 아래와 같은 메시지를 보낸다.

GSC → GSCA : {GK, IDG, {token_GSC} prv_GSC} pbk_GSCA

GK : 그룹키
GSC(Group Security Control) : GSC 신분정보
IDG : GSC ID
prv_GSC : GSC의 개인키
pbk_GSCA : GSCA의 공개키

그룹키와 GSC의 ID, 개인키로 전자서명한 GSC 신분정보를 GSCA의 공개키로 암호화하여 GSCA로 전송한다.

이 메시지를 받은 GSCA는 각각 자신의 개인키로 복호화하여 그룹 키를 얻으며 token_GSC의 전자서명값을 검증하여 GSC의 신분을 확인한다. GSCA는 다시 GSC에게 아래와 같은 응답메시지를 보낸다.

GSCA → GSC : {IDG, IDGSCA, {token_GSCA} prv_GSCA} pbk_GSC

IDG : GSC ID
GSCA(Group Security Control Agent) : GSCA 신분정보
IDGSCA : GSCA ID
prv_GSCA : GSCA의 개인키
pbk_GSC : GSC의 공개키

이는 IDG, IDGSCA, 전자서명한 GSCA 정보를 GSC의 공개키로 암호화하여 GSC로 보내주게 되며, GSC는 GSCA의 응답메시지를 받으면 자신의 개인키로 복호화한다. 다음 GSCA의 공개키로 token_GSCA를 복호화하여 GSCA의 신분을 확인하고 그룹 키가 안전하게 각 GSCA에게 전달되었는지를 확인한다. GSCA는 그룹키를 받고 또 자신의 서버그룹 키 DK를 안전하게 각 멤버에게 전송해야 한다. 여기서 DK는 위에서 설명한 것과 마찬가지 방법으로 각 멤버에게 전송한다. 이렇게 그룹 키 생성과 분배과정이 완성되면 다음 단계로 들어간다.

3.3 키 갱신과정

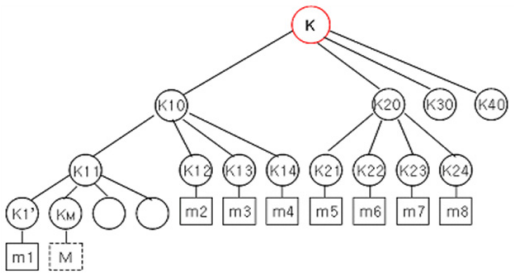
본 논문에서는 그룹 키와 서브그룹 키, 두 가지 방면의 키 갱신이 있다. 여기서 GSCA는 자신이 관리하는 서브그룹내의 멤버의 변화에 따라 서브그룹 키를 갱신하고 GSC는 그룹 키의 생명주기가 끝나거나 멀티캐스트의 요구에 따라 GSCA를 추가하거나 탈퇴시킬 때 그룹 키를 갱신한다.

3.3.1 멤버 가입

한 사용자가 멀티캐스트 그룹에 가입을 원한다면 자신과 가장 가까운 곳에 위치하고 있는 GSCA에게 가입메시지 JOIN_REQUEST를 보낸다. GSCA는 이 가입메시지를 받으면 공인인증기관 PKA(Public Key Authority)에 가입 멤버의 공개키 인증서를 요청하여 받는다. 그리고 GSCA는 등록모듈을 시동하여 사용자의 가입요청을 처리하고 사용자 아이디나 공개키 등 정보를 DB에 저장한다. 다음으로는 서브그룹 키 갱신이 이루어진다. 본 논문은 4진 트리가 전면적으로 키에 대한 비교적 높은 성능을 가진다는 것을 통하여[5,6] 논리적인 4진 키 트리를 구성한다. GSCA와 그가 관리하는 각 멤버들 사이에는 일방향 해쉬함수 H를 공유하고 있다. 즉 K'=H(K, R)를 이용하여 키

갱신을 한다. 한 GSCA에 16개 멤버가 있는 키 트리구조를 예로 들어 설명한다.

그림 5는 한 멤버가 가입을 했을 때 키 갱신을 보여준다.



[그림 5] 멤버 가입 시 GSCA의 키 갱신

여기서 멤버 m1에서 m16은 각각 트리구조에서 끝부분의 대응되는 노드 키를 가지고 있으며 루트노드에 대응되는 키는 서브그룹 키 DK이다. GSCA는 트리상의 모든 키를 가지고 있으며 각 멤버들은 자신이 위치하고 있는 끝부분 노드로부터 루트노드까지 경로상의 모든 키를 가지고 있다. 한 사용자 M이 가입을 요청하였을 때 Backword Secrecy를 보장하기 위하여 서브그룹 키를 갱신할 필요가 있다. 이때 GSCA는 난수 R을 생성한 후 해쉬 함수를 이용하여 다음 그룹키를 생성하여 유니캐스트 방식으로 새롭게 생성된 키를 M에게 보낸다. 전송시에는 GSCA의 개인키를 이용하여 전자서명을 한 후, M의 공개키를 이용하여 암호화한 후 전송하게 된다.

GSCA → M : {new_DK, token_GSCA} prv_GSCA} pbk_M

new_DK : 새로운 서브 그룹 키
token_GSCA : GSCA 신분정보
prv_DSCA : GSCA의 개인키
pbk_M : M의 공개키

GSCA는 키 트리 구조 끝부분에 하나의 새로운 노드를 만들어 M을 대응시키고 Participant_List에 멤버를 추가한다. 다음은 전 서브 그룹키로 난수 R로 암호화하여 그룹의 나머지 멤버들에게 멀티캐스트로 전송한다.

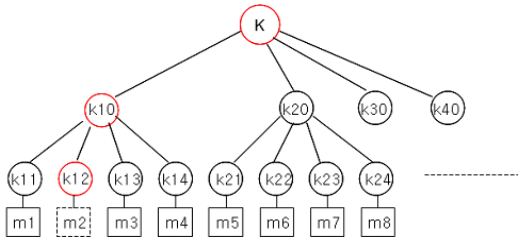
GSCA → other member : {R, token_GSCA} prv_GSCA} old_DK

R : 난수
token_GSCA : GSCA 신분정보
prv_DSCA : GSCA의 개인키
old_DK : 기존에 사용한 그룹키로 암호화

나머지 멤버들은 GSCA의 메시지를 받고 old_DK로 복호화하여 파라미터 R을 얻는다. 그리고 해쉬함수 H를 이용하여 새로운 서브그룹 키 new_DK를 생성한다.

3.3.2 멤버 탈퇴

한 멤버가 그룹을 탈퇴하기를 원한다면 GSCA에게 그룹 탈퇴 메시지 LEAVE_REQUEST를 보낸다. GSCA는 Participant_List에서 탈퇴를 원하는 멤버를 삭제한다. 역시 Forward Secrecy를 보장하기 위하여 GSCA는 서브그룹 키를 갱신할 필요가 있다. 그림 6은 멤버 탈퇴 시 키 갱신을 보여준다.



[그림 6] 멤버 탈퇴 시 GSCA의 키 갱신

m2가 그룹을 탈퇴한다고 가정했을 때 GSCA는 m2가 위치한 끝부분 노드로부터 루트노드까지 경로상의 모든 키를 갱신해야 한다. GSCA는 랜덤으로 난수 R을 생성하여 나머지 멤버들에게 아래와 같이 전송한다.

GSCA -> m1, m3, m4 : {R}k11, {R}k13, {R}k14
GSCA -> m5～m8 : {R}k20
GSCA -> m9～m12 : {R}k30
GSCA -> m13～m16 : {R}k40

그러면 m2를 제외한 모든 서브그룹 멤버는 R를 받는다. 다음 각 멤버는 역시 해쉬함수를 이용하여 전 서브그룹 키와 R로 새로운 서브그룹 키를 생성한다. 여기서 m1, m3, m4는 서브그룹 키뿐만 아니라 경로상의 k10도 갱신해야 한다. 이는 역시 해쉬함수를 사용하여 갱신할 수 있다.

3.3.3 주기적인 그룹 키 갱신

그룹 키는 모두 유한된 생명주기를 갖고 있기에 주기적으로 갱신을 해야 한다. 그룹 키 갱신은 서브그룹 키 갱신과 달리 주로 GSC와 GSCA간에 이루어진다. GSC는 그룹 키 생명주기가 끝나거나 멀티캐스트의 요구에 따라 GSCA를 추가 및 탈퇴시킬 때 그룹 키를 갱신한다.

Signed token에 포함된 타임스탬프를 이용하여 그룹 키의 생성시간을 기록하고 키의 주기인 Timer값을 설정한다. 이 Timer값은 일정한 값으로 쓰거나 임의의 값으로 매번 다르게 설정할 수 있다. 정해진 Timer값을 지나면 GSC는 타임스템프를 이용하여 그룹 키의 끝나는 시간을 기록하고 순방향 키 체인기법을 난수 R값 즉 Nonce에 따라 GSC 그룹 키를 생성하게 되며, 시간이 만료되면 Timer 값을 다시 설정한다. 각 GSCA는 타임스탬프 값을 확인하고 역시 키 체인기법을 난수 R에 따라 다음 주기에 사용될 그룹 키를 만든다. 여기서 Nonce값은 Signed token에 포함된 랜덤 수를 이용한다. 그림 7은 GSC와 GSCA 간의 그룹 키 갱신을 보여준다.

[그림 7] GSC와 GSCA간의 그룹 키 갱신

4. 성능평가

키 관리기법의 성능을 평가하기 위해서는 크게 보안성과 효율성으로 분석한다. 본 논문에서는 주로 Rekey 작업을 위한 Storage 비용, Communication 비용, Computation 비용의 시스템 효율성에 대한 분석을 통하여 성능평가를 진행한다. 여기서 제안된 기법과 Iolus 기법의 각 하위 그룹들은 서로 독립적으로 이루어지기 때문에 최하위 그룹(Iolus: GSI, 제안 기법: GSCA)만을 기준으로 LKH 기법과 비교 분석한다.

4.1 Storage cost 분석

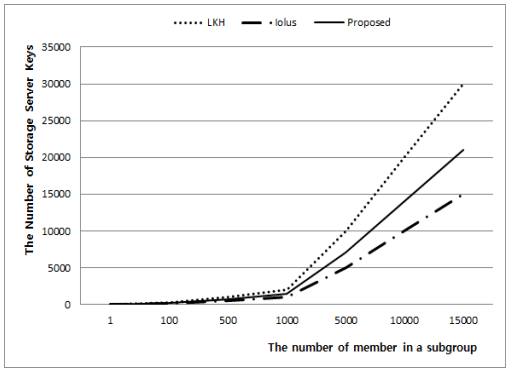
Storage 비용은 주로 그룹 관리서버와 멤버들이 관리하는 키의 수를 가지고 분석한다. Star형방식과 같은 단순 그룹 키 관리기법 일 경우 키의 개수가 가장 적겠지만 전체적인 시스템 측면에서는 매우 비효율적이라고 볼 수 있다. 즉, 이런 기법은 장애 발생 시, 장애 복구 시, Rekey 작업 시 등 면에서 트리기반의 계층형기법보다 비효율적이다. Iolus기법도 역시 최하위 그룹에서는 단순 그룹 키 관리기법을 적용하였기에 그 효과는 동일하다. 아래 표 1은 각 그룹키 관리기법의 키 저장 개수를 나타낸다. 서버가 N개의 멤버를 관리(한 서브 그룹내의 멤버수)하고, 트리의 높이를 h로, 트리의 지수를 r로 가정했을 때 아래와 같은 식을 얻을 수 있다.

$$(N = r^{h-1},\ h = Log_r N+1)$$

[표 1] Storage 비용 비교

키 관리기법	키 개수	
	서버	멤버
LKH	$2N-1$	$Log_2 N+1$
Iolus	$N+1$	2
제안하는 기법	$\frac{4N-1}{3}$	$\frac{1}{2} Log_2 N+1$

LKH 기법에서 서버는 이진 트리구조상의 모든 키를 갖고 있어야 함으로 $2N-1$개의 키를 유지하고 있다. 멤버는 2진 트리에서 자신과 연결된 최하위 노드의 키로부터 루트노드의 경로상의 모든 키를 갖고 있어야 함으로 $Log_2 N+1$개의 키를 유지하고 있다. 제안하는 키 관리기법은 4진 트리로 구성 되었기에 k=4이다. 서버는 $\frac{4N-1}{3}$개의 키를 유지하며 멤버는 $\frac{1}{2} Log_2 N+1$개의 키를 유지한다. 따라서 LKH 기법보다 트리의 높이가 작으며 서버와 멤버의 키 저장비용도 적다는 것을 알 수 있다. 이는 그림 8과 그림 9를 통해 그 차이를 알 수 있다.



[그림 8] 서버의 키 개수

[그림 9] 멤버의 키 개수

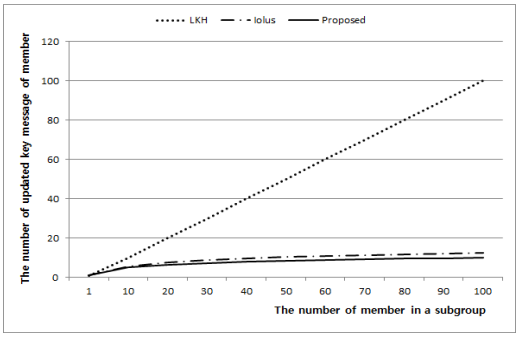
4.2 Communication 비용 분석

Communication 비용은 주로 키 갱신에 필요한 메시지에 대한 비용으로 멤버의 가입과 탈퇴 시에 부족한 대역폭에 대한 부하를 줄이기 위해서는 메시지 수를 줄여야 한다. 아래 표 2는 LKH 기법, Iolus 기법과 제안하는 키 관리기법의 Communication 비용을 비교한 것이다.

[표 2] Communication 비용 비교

키 관리기법	키 갱신 메시지 수	
	멤버 가입 시	멤버 탈퇴 시
LKH	Log_2N+1	$2Log_2N-1$
Iolus	2	$N-1$
제안하는 기법	2	$\frac{3}{2}Log_2N$

Communication 비용은 멤버의 가입과 탈퇴 시 두 가지 경우로 분석한다. 멤버가 가입할 경우 LKH 기법은 서버가 모든 키를 관리하는 이진 트리구조로서 키 갱신 시 그룹 키뿐만 아니라 가입된 멤버의 트리상의 모든 키를 갱신한다. 따라서 서버는 키 갱신에 필요한 메시지 수가 트리의 높이만큼 필요하다. 하지만 Iolus 기법과 제안하는 키 관리기법은 단 한 번의 멀티캐스트와 한 번의 유니캐스트로 키 갱신이 완료됨으로 메시지 수는 2개가 된다. 멤버 탈퇴 시 LKH 기법은 서버가 탈퇴한 멤버가 갖고 있는 모든 키를 갱신하고 분배를 한다. 따라서 키 갱신 메시지 수는 $2Log_2N-1$개가 필요하다. Iolus 기법은 멤버를 Star형 방식으로 관리하기 때문에 멤버의 수만큼 멤버들의 개별 키로 암호화하여 유니캐스트로 전송해야 한다. 때문에 $N-1$개의 키 갱신 메시지 수가 필요하다. 하지만 제안하는 키 관리기법은 키 갱신 시 서버는 탈퇴한 멤버가 갖고 있는 트리상의 모든 키를 갱신해서 각 멤버에게 보내는 것이 아니라 단지 키 갱신에 필요한 파라미터 R 값만 보내면 된다. 따라서 그림 10과 같이 키 갱신에 필요한 메시지 수는 $\frac{3}{2}Log_2N$개다.



[그림 10] 멤버의 갱신 키에 따른 메시지 수

4.3 Computation 비용 분석

Computation 비용 분석은 멤버의 변화에 따른 그룹 키 갱신에 필요한 계산비용을 비교한다. 즉 멤버 가입과 탈퇴 시 새로운 키 생성에 필요한 시간과 키 분배를 위한 암복호화에 소비되는 시간을 비교하는 대상으로 한다. 제안하는 키 관리기법의 경우 일방향 함수를 수행하는 시간이 포함된다. 아래 표 3은 LKH 기법, Iolus 기법과 제안하는 키 관리기법의 Computation 비용을 비교한 것이다.

[표 3] Computation 비용 비교

키 관리기법	계산 비용	
	서버	멤버
LKH	$2(Log_2N+1)T_e$	$(\log_2N+1)T_d$
Iolus	nT_e	T_d
제안하는 기법	$(\frac{1}{2}Log_2N)T_h$ $+(\frac{3}{2}Log_2N)T_e$	$(\frac{1}{2}Log_2N)T_h+T_d$

여기서 T_e는 키 분배를 위한 키 암호화에 소비되는 시간이다. T_d는 암호화된 키를 복호화에 필요한 시간을 의미하며 T_h는 일방향 해쉬함수를 수행하여 새로운 키를 생성하는 시간이다. LKH 기법에서는 한 멤버가 가입/탈퇴 할 경우 그룹 관리서버가 모든 키를 생성 및 분배하기 때문에 Log_2N+1개의 Rekey를 전송해야 한다. 따라서 그룹 관리서버의 계산비용은 $2(Log_2N+1)T_e$이다. 멤버의 경우 역시 트리 높이 수만큼의 Rekey를 복호화하는 시간이 필요함으로 그 계산비용은 $(\log_2N+1)T_d$이다. Iolus 기법에서 멤버의 가입/탈퇴로 인하여 그룹 관리서버는 N번의 새로운 키를 암호화해서 보내야 하고 멤버는 단 한 번의 복호화하는 시간이 필요하다. 이 기법에서 멤버의 계산시간은 적으나 그룹 관리서버의 계산시간은 그룹 멤버의 수만큼 증가되기 때문에 작업처리 시간이 많이 소요된다. 제안하는 키 관리기법은 한 멤버의 가입/탈퇴 시 오직 키 생성에 필요한 파라미트 R만 분배하고 나머지 갱신해야 할 키들은 일방향 함수를 이용하여 생성한다. 따라서 n명의 멤버를 가진 GSCA는 한 명의 멤버가 가입/탈퇴 시 $(\frac{3}{2}Log_2N)T_e$의 암호화 시간과 $(\frac{1}{2}Log_2N)T_h$의 키 생성 시간이 필요하다. 멤버도 역시 한번은 복호화와 $(\frac{1}{2}Log_2N)T_h$의 해쉬함수를 이용한 키 생성 시간이 필요하다.

그리고 윈도우 환경에서 256Bit 키 길이를 이용하여 실험평가를 한 결과 RSA암호화 알고리즘을 사용하여 키 암호화하는데 필요한 시간은 약 91μs이며 그 키를 다시 복호화하는데 필요한 시간은 약 113μs이다. 하지만 해쉬 함수 MD5를 이용하여 새로운 키를 생성하는 시간은 약 1μs이다. $T_h << T_e$, $T_h << T_d$임으로 제안하는 그룹 키 관리기법은 LKH 기법이나 Iolus 기법보다 Rekey 작업을 위한 처리시간 면에서 보다 우수하다.

제안하는 키 관리기법은 Iolus 기법보다 키의 저장량이 많지만 키 갱신을 위한 메시지 수나 키 갱신에 필요한 처리시간 면에서 다른 기법보다 적다는 것을 볼 수 있다. 따라서 종합적인 효율성을 분석하여 봤을 때 제안하는 키 관리기법이 보다 우수하다.

5. 결론

본 논문에서 제안한 그룹 키 관리기법은 멀티캐스트 환경에 적합하며 에이전트 기술을 도입하여 서브그룹을 관리함으로써 서버의 부하를 줄인다. 또한 GSCA내의 효율적인 그룹 키 생성 및 분배를 위하여 일방향 해쉬함수를 이용한 트리기반의 그룹 키 관리 알고리즘을 적용한다. 이렇게 함으로써 멤버의 변화가 빈번한 멀티캐스트 환경에서 키 갱신 메시지 수를 줄이고 키 분배에 대한 지연을 감소한다. 또한 Signed token을 추가함으로써 현존하는 그룹 키 관리기법의 멤버인증의 취약점을 보완한다. 이를 통해 서브 그룹 분리를 통한 그룹 키의 키 갱신에 대한 분배 시 전송 시간과 키의 암호화 생성 시간이 기존의 키 관리 시스템 보다 빠르다. 또한 적은 향의 스토리지를 사용함으로써, 비용을 절감할 수 있고 Singed token으로 인한 그룹 내 멤버 인증의 취약점을 해결할 수 있었다.

향후 연구과제로는 먼저 본 논문에서 제안한 시스템을 직접 구현하여 기존의 기법과 비교·분석하는 것이다. 또한 GSC와 GSCA간의 주기적인 그룹 키 갱신을 위한 세부적인 Timer 설정 알고리즘이 필요하며 세부적인 GSCA 선출기법이 필요할 것이다. 또한 다수의 에이전트 운용 시, 에이전트의 운용에 대한 추가 비용을 해결하고, 에이전트 마비 시 해당 서브그룹에 대한 멤버의 통신 문제를 해결해야 할 것이다.

참고문헌

[1] B. Cain, S. Deering, I. Kouvelas, B. Fenner, A. Thyagarajan, "internet Group Management Protocol, version3," RFC3376, Oct. 2002.

[2] T. Ballardie, J. Crowcroft, "Multicast-Specific Security Threats an Counter-Measures," Proceedings of the Symposium on Network and Distributed System Security, Feb. 1995.

[3] Hugh Harney, Eric Harder, “LKH : Logical Key Hierarchy Protocol,” Internet Draft, draft-harney-sparta-lkhp-sec-00.txt, Mar. 1999.

[4] Suvo Mittra, "Iolus : A Framework for Scalable Secure Multicasting," Proceeding of the ACM SIGCOMM '97, Sep. 1997.

[5] Canetti R, Caray J, Itkis G, et al, "Multicast security: a taxonomy and some efficient constructions", Proc of the INFOCOM’99, 708- 716, 1999.

[6] Chung Kei Wong, Mohamed Gouda, Simon S.Lam, "Secure group communication using key graphs," IEEE/ACM Transactions on networking, VOL.8, NO.1, Feb. 2000.

[7] Alan T. Sherman and David A. McGrew, "Key Establishment in Large Dynamic Groups using One Way Function Trees," IEEE Trans. on Software Engineering, Vol.29, No.5, pp. 444-458, 2003.

[8] 장봉덕, “동적 멀티캐스트 환경에서 에이전트기반의 분산형 그룹키 관리기법,” 숭실대학교 석사학위논문, 2월, 2009.

김 보 승(Bo-Seung Kim) **[정회원]**

- 2002년 2월 : 영동대학교 컴퓨터공학과 공학사
- 2004년 8월 : 숭실대학교 컴퓨터학과 공학석사
- 2005년 3월 ~ 현재 : 숭실대학교 컴퓨터학과 박사과정

<관심분야>
멀티캐스트, IPTV, 센서 네트워크, IPv6, DNS, 홈네트워크

김 정 재(Jung-Jae Kim) **[정회원]**

- 1999년 2월 : 영동대학교 컴퓨터공학과 공학사
- 2001년 2월 : 숭실대학교 컴퓨터학과 공학석사
- 2005년 8월 : 숭실대학교 컴퓨터학과 공학박사

<관심분야>
암호학, DRM, RFID/USN, 네트워크 보안

장 봉 덕(Feng-De Zhang) **[정회원]**

- 2003년 6월 : 연변과학기술대학교 계산기학과 공학사
- 2009년 2월 : 숭실대학교 컴퓨터학과 공학석사
- 2009년 3월 ~ 현재 : 에이치텔레콤(주) S/W 전임연구원

<관심분야>
멀티캐스트, 센서 네트워크, 홈네트워크, 모바일 통신

신 용 태(Yong-Tae Shin) **[정회원]**

- 1985년 2월 : 한양대학교 산업공학과 공학사
- 1990년 12월 : Iowa대학교 전산학과 공학석사
- 1994년 5월 : Iowa대학교 전산학과 공학박사
- 1995년 3월 ~ 현재 : 숭실대학교 컴퓨터학부 교수

<관심분야>
멀티캐스트, IPTV, IPv6, RFID/USN, 네트워크 보안