접근 제어를 위한 반응적 방식의 그룹키 관리 기법
(Group Key Management Scheme for Access Control with Reactive Approach)

김 희 열 † 이 윤 호 †† 박 용 수 ††† 윤 현 수 ††††
(Heeyoul Kim) (Younho Lee) (Yongsu Park) (Hyunsoo Yoon)

요 약 다양한 종류의 데이타 스트림과 다양한 권한을 가지는 사용자들을 위한 그룹 통신을 위해서는 접근 제어(access control)가 필수적이다. 동일한 접근 권한을 가지는 그룹 멤버들은 하나의 클래스에 속하게 되며, 이러한 클래스들은 주어진 접근 관계를 표현한 하나의 계층을 구성한다. 그리고 각 클래스에는 하나의 비밀키가 할당된다. 기존의 기법들에서는 계층으로부터 하나의 논리적 키 트리를 생성하고 각 사용자는 항상 자신이 접근할 수 있는 모든 클래스의 키를 관리하는 방식, 즉 선행적(proactive)인 방식이었다. 하지만, 계층의 규모가 큰 경우에 사용자가 키를 저장하기 위한 공간이 늘어나고 키 갱신을 위한 메시지 또한 커진다는 단점을 가진다. 그리고 대부분의 경우 사용자는 극히 일부의 스트림만을 동시에 접근하게 되며, 이를 위해 모든 키를 지속적으로 갱신하는 것은 낭비가 된다. 본 논문에서는 이를 고려한 반응적(reactive)인 방식의 키 관리 기법을 제안한다. 각 사용자는 자신이 속한 서브그룹의 키만을 관리하며 다른 키가 필요한 경우에만 자신의 키와 공개 파라미터를 이용해서 해당 키를 추출하게 된다. 이로 인해 키 갱신을 위한 비용이 줄어들게 되고, 특히 접근 관계가 복잡하고 규모가 큰 그룹에 대해 좋은 성능을 가진다. 그리고 접근 관계가 변하는 경우, 이를 쉽게 반영할 수 있다는 장점을 가진다.

키워드 : 그룹통신, 접근 제어, 키 관리

Abstract In the group communication which has multiple data streams and various access privileges, it is necessary to provide group access control. The group members having the same access privilege are classified into one class, and the classes form a hierarchy based on the access relations. Then each class is assigned to a secret key. In the previous schemes, a single logical key graph is constructed from the hierarchy and each member always holds all secret keys of the classes he can access in the proactive manner. Thus, higher-privileged members hold more keys then lower-privileged members. However, if the hierarchy is large, each member manages too many keys and the size of multicast message in rekeying increases in proportion to the size of the hierarchy. Moreover, most of the members access a small portion of multiple data streams simultaneously. Therefore, it is redundant to receive rekeying message and update the keys in which he is not currently interested. In this paper, we present a new key management scheme that takes a reactive approach in which each member obtains the key of a data stream only when he wants to access the stream. Each member holds and updates only the key of the class he belongs. If he wants to get the key of other class, he derives it from his key and the public parameter. Proposed scheme considerably reduces the costs for rekeying, especially in the group where access relations are very complex and the hierarchy is large. Moreover, the scheme has another advantage that it easily reflects the change of access relations.

Key words : group communication, access control, key management

· 이 논문은 2005, 2006년 정부(교육인적자원부)의 재원으로 한국학술진흥재단의 지원을 받아 수행된 연구임(KRF-2005-042-D00294, KRF-2006-331-D00556)

† 비 회 원 : 한국과학기술원 전산학과
hykim@nslab.kaist.ac.kr
†† 비 회 원 : 한국과학기술원 전산학과 BK 포닥
yhlee@nslab.kaist.ac.kr
††† 종신회원 : 한양대학교 컴퓨터공학 교수
yongsu@hanyang.ac.kr
(Corresponding author임)
†††† 종신회원 : 한국과학기술원 전산학과 교수
hyoon@nslab.kaist.ac.kr

논문접수 : 2007년 1월 22일
심사완료 : 2007년 7월 10일

정보과학회논문지: 시스템 및 이론 제34권 제11호(2007.12)

1. 서 론

최근 화상 회의, 인터넷 방송, DMB등 대규모의 데이타를 여러 사용자에게 동시에 전송하는 애플리케이션이 증가함으로써 그룹 통신의 중요성이 부각되고 있다. 이러한 애플리케이션의 보안 문제와 상업화를 위해서는 허가된 그룹 멤버만이 데이타를 접근할 수 있게 하는 안전한 그룹 통신을 기반으로 해야 하며, 이는 적합한 그룹 멤버들만이 공유하는 그룹키를 통해 데이타를 암호화해서 전송함으로써 해결된다. 여러 사용자가 참여하는 그룹 통신의 특성상 그룹 멤버쉽은 매우 빈번하게 변하게 되며, 이러한 경우에도 데이타의 기밀성이 유지되도록 후방위 안전성(backward secrecy)과 전방위 안전성(forward secrecy)을 보장해야 한다. 이를 위해서 멤버쉽의 변화가 생길 때마다 그룹키를 갱신해야 하며, 규모가 큰 그룹에 대해서도 효율적으로 키 갱신을 하기 위한 그룹키 관리 기법들이 연구되어 왔다[1-5].

기존의 그룹키 관리 기법은 하나의 데이타 스트림에 대한 동일한 접근 권한만을 고려하고 있다. 하지만, 실제적으로 많은 그룹 통신 애플리케이션은 여러 종류의 데이타 스트림을 가지며, 각 멤버도 다양한 접근 권한을 가진다. 예를 들어, 인터넷 방송의 경우에 스포츠 중계, 뉴스, 날씨 등 다양한 채널이 존재하며, 각 채널에 대한 접근 권한도 사용자별로 다양하게 된다. 이를 위해서는 서로 다른 권한을 가지는 사용자들을 위한 접근 제어가 요구된다.

접근 제어를 위해서는 우선 동일한 접근 관계를 가지는 사용자들을 하나의 클래스로 묶고, 마찬가지로 동일한 관계를 가지는 데이타 스트림도 하나의 클래스로 묶는다. 그리고 클래스간의 관계를 고려해서 unified 계층을 생성하고[6], 이를 바탕으로 키를 분배하게 된다. 가장 기본적인 방법은 각 클래스에 독립적인 키를 할당한 후 각각의 사용자는 자신이 속한 클래스와 모든 접근 가능한 클래스의 키를 소유하는 것이다. 하지만 이는 계층이 복잡해질수록 많은 저장 공간을 필요로 한다는 단점을 가지며, 이를 해결하기 위해 기존의 계층적 접근 제어(Hierarchical Access Control: HAC) 분야에서 다양한 접근 방법이 제안되어 왔다[7-9]. 하지만, 이러한 연구들은 그룹 통신을 대상으로 한 것이 아니라서 그룹 멤버쉽 변화에 따른 키 갱신을 고려하지 않고 있다.

최근에 그룹 통신에서의 접근 제어를 위한 키 관리 기법들이 제안되었다. 이 중 Chinese Remainder Theorem에 기반한 기법은 사용자가 트리 형태의 계층을 가지는 경우에만 적용될 수 있다는 한계를 가진다[10]. [11]와 [12]는 기존의 단일 그룹을 위한 키 관리 기법인 논리적 키 트리(logical key tree)를 확장하는 기법을 제안했으며, 위의 기본적인 방법인 사용자가 모든 접근 가능한 클래스의 키를 소유하는 방식을 따른다. 이러한 기법들은 모두 선행적(proactive) 방법, 즉 자신이 접근할 수 있는 모든 키에 대해 키 갱신 과정을 항상 수행하다가 원하는 데이타 스트림을 키들 중 하나로 얻는 방법이다. 이로 인해 접근 관계가 복잡해질수록 사용자가 저장해야 할 키의 개수가 많아지며, 또한 키 갱신시에 필요한 메시지의 크기가 크게 늘어나게 된다. 하지만, 대부분의 경우에 사용자는 여러 데이타 스트림중 일부분의 스트림만을 동시에 접근하게 된다. 위의 예에서 사용자는 여러 채널 중 대부분 하나의 채널만을 시청하며, 아무 채널도 시청하지 않는 경우도 상당히 많다.

이 논문에서는 이에 대한 대안으로 반응적(reactive)인 방식, 즉 사용자가 원하는 데이타 스트림을 접근하려 할 때 이에 대한 키를 획득하는 방식의 키 관리 기법을 제안한다. 각 사용자는 자신이 속한 클래스의 키만을 소유하며, 다른 클래스의 키는 자신 클래스의 키와 공개되어 있는 파라미터를 통해 획득할 수 있는 방식으로 이루어진다. 이 방식은 접근 관계가 복잡해지더라도 사용자가 부담해야 하는 오버헤드가 적다는 장점을 가진다. 자신이 속한 클래스의 멤버에 변화가 있을 때에는 전송되는 메시지를 통해 키를 갱신해야 하지만, 다른 클래스의 멤버에 변화가 있을 때에는 부가적인 정보 없이 쉽게 키를 갱신할 수 있게 된다. 또한, 제안된 기법은 접근 관계에 변화가 있는 경우에도 이를 쉽게 반영할 수 있다는 장점을 가진다.

본 논문의 구성은 다음과 같다. 2장에서는 그룹 통신에서의 엑세스 제어를 위한 기본적인 방법과 기존 연구를 살펴보고 개선할 부분을 설명한다. 3장에서는 제안하는 반응적 방식의 키 관리 기법을 설명한다. 4장에서는 제안된 기법의 성능을 분석하고 5장에서는 안전성을 분석한다. 그리고 6장에서 결론을 맺는다.

2. 관련 연구

이번 장에서는 우선 그룹 통신에서 접근 제어를 하기 위해 주어진 접근 관계로부터 계층을 구성하는 방법에 대해 살펴보겠다. 그리고 이를 기반으로 한 가장 최근의 기법인 HAC 키 그래프를 살펴보고 장단점을 논의하겠다.

2.1 그룹 통신에서의 접근 제어

그룹 내의 모든 사용자는 동일한 접근 권한을 가지는 멤버들끼리 하나의 유저 서브그룹을 형성하며, 각 멤버는 이러한 유저 서브그룹 $U_1, ..., U_l$중 하나에 속하게 되고 접근 관계를 바탕으로 각 서브그룹은 관계 "$\leq$"에 의해 부분적으로 정렬(partially ordered)된다. 이때 $U_i \leq U_j$는 U_j에 속한 사용자는 U_i에 속한 사용자가 접근하는 모든 데이타 스트림을 접근할 수 있다는 것을 의미한다. 마찬가지로 모든 데이타 스트림은 자신을 접

근할 수 있는 사용자의 집합이 동일한 데이타 스트림끼리 하나의 데이타 서브그룹을 형성하며, 각 데이타 스트림은 이러한 데이타 서브그룹 $D_1,...,D_m$중 하나에 속하게 되고 각 서브그룹은 관계 "≤"에 의해 부분적으로 정렬된다. 이때 $D_i \leq D_j$는 D_j에 속한 데이타 스트림을 접근할 수 있는 사용자의 집합이 D_i에 속한 데이타 스트림을 접근할 수 있는 사용자 집합의 부분집합임을 의미한다.

이렇게 형성된 유저 서브그룹의 계층과 데이타 서브그룹의 계층을 통합해서 통합 계층을 만들 수 있다. 그리고 이는 유향 비순환 그래프(Directed Acyclic Graph: DAG)의 형태로 표현이 된다. 이때 각 노드에는 하나의 유저 서브그룹, 또는 하나의 데이타 서브그룹, 또는 하나의 유저 서브그룹과 하나의 데이타 서브그룹이 속하게 된다. 그림 1에서 노드 V_1에는 유저 서브그룹 U_1이 속하고, 노드 V_2에는 유저 서브그룹 U_2와 데이타 서브그룹 D_2가 속하게 된다.

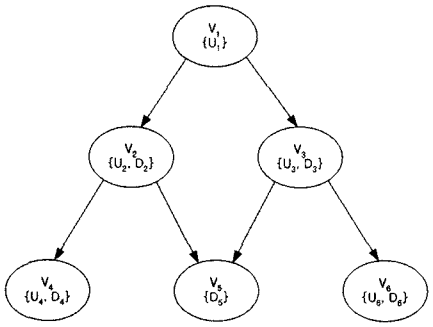

그림 1 계층적 접근 제어를 위한 유향 비순환 그래프

생성된 DAG의 각 노드들 역시 관계 "≤"에 의해 부분적으로 정렬된다. V_i로부터 V_j로 가는 패스가 존재할 때 $V_j \leq V_i$라고 하며, 이는 U_i에 속한 사용자가 D_j에 속한 데이타 스트림을 접근할 수 있음을 의미한다. 이때 V_i는 V_j의 선행자(predecessor)라고 하며, V_j는 후계자(successor)라고 한다.

접근 제어를 위한 가장 기본적인 키 관리 방법은 각 노드 V_i마다 하나의 비밀키 K_i를 선정하는 것이다. 그리고 각 사용자는 자신의 유저 서브그룹이 속한 노드와 모든 후계자 노드의 키를 가지고, 각 데이타 스트림은 자신의 데이타 서브그룹이 속한 노드의 비밀키로 암호화되어서 전송된다. 예를 들어, U_2에 속한 사용자는 $\{K_2, K_4, K_5\}$를 가지게 되며 각각의 키로 암호화된 데이타 스트림 $\{D_2, D_4, D_5\}$를 접근할 수 있게 된다. 반면에 U_4에 속한 사용자는 오직 $\{K_4\}$만을 가지기 때문에 D_4를 제외한 다른 데이타 스트림을 접근할 수 없다.

하지만, 이러한 방법은 각 사용자가 저장해야 하는 키의 개수가 최대 DAG의 노드 수만큼 된다는 단점을 가진다. 그리고 사용자의 멤버쉽 변화가 있을 경우, 일부 키들을 갱신할 수 있는 방안이 제공되지 못한다.

2.2 HAC 키 그래프

[11]에서는 기존의 단일 그룹을 위한 키 갱신 방법인 논리적 키 트리를 확장해서 접근 제어를 수행하는 방식을 제안했다. 우선 각 유저 서브그룹 U_i별로 논리적 키 트리를 구성하고 루트 노드의 키 SK_i를 서브그룹 관리를 위한 서브그룹키로 한다. 그리고 이 트리를 DAG의 각 노드의 하위에 연결시킨다. 그리고 DAG를 일부 서브트리가 공유되는 형태의 여러 개의 이진 트리로 변환한다. 그림 2는 그림 1을 변환한 키 그래프를 나타낸다.

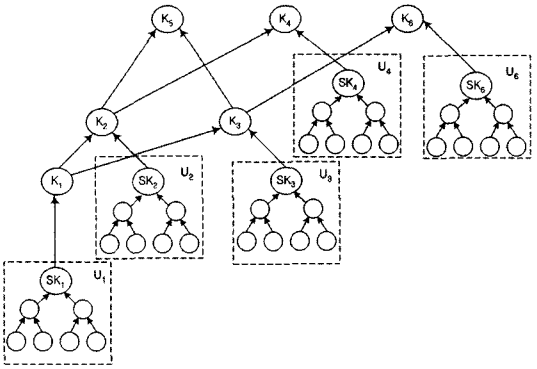

그림 2 HAC 키 그래프

각 사용자는 하나의 종단(leaf) 노드에 대응하며, 그 종단 노드로부터 각각의 루트 노드까지의 패스상에 있는 모든 키들을 소유하게 된다. 그리고 한 사용자가 그룹을 탈퇴하는 경우, 그 사용자가 소유하고 있던 모든 키가 바뀌게 된다. 예를 들어, U_2에 속한 사용자가 탈퇴하면 서브그룹키 SK_2를 포함한 서브그룹 내에서 사용자가 알고 있던 키가 갱신되며, 마찬가지로 U_2의 멤버로써 알 수 있는 키들인 $\{K_2, K_4, K_5\}$가 갱신된다. 키를 갱신하는 방법은 갱신되는 키의 노드의 두 자식 노드의 키로 암호화를 해서 멀티캐스팅되는 방식으로, 이는 갱신되어야 하는 키의 개수가 d일 때, $2d$만큼의 암호화된 키가 전송된다.

이러한 방식의 접근방법은 각 데이타 스트림별로 별도의 키 트리를 구성하는 것보다 저장해야 하는 키의 개수가 줄고 키 갱신시에도 더 적은 메시지를 요구한다는 장점을 가진다. 하지만, 이는 몇 가지 개선할 사항을 가지고 있다.

- 각 사용자가 저장해야 하는 키의 최대 개수가 DAG의 노드 수에 비례한다. 기존의 동일한 접근 권리를 가지는 그룹 통신을 위한 키 트리의 경우, 전체 사용자의 수가 1,000,000명으로 매우 많더라도 각 사용자는 $\log 1000000 = 20$개의 키만을 저장했다. 하지만 위의 방법에서는 균형적인 트리가 되지 않으며, 서브그룹을 위한 키들을 제외하더라도 각 사용자는 자신의 노드와 모든 후계자 노드의 키를 모두 저장해야 한다.
- 키 갱신을 위한 메시지의 크기가 증가한다. 저장해야 하는 키의 개수와 마찬가지로 키 갱신시 필요한 메시지의 크기도 최대 DAG의 노드 수만큼 증가하게 된다. 특히, 통신량이 그룹 통신의 가장 큰 장애가 된다는 점에서[13] 증가하는 메시지의 크기는 확장성(scalability)에 많은 영향을 준다.
- 접근 관계의 변화를 고려하고 있지 않다. 접근 관계의 변화는 DAG에서 노드가 추가/제거되거나 두 노드 사이의 에지가 추가/제거되는 것으로, 애플리케이션의 특성과 정책의 변화에 따라 변할 수 있다. 하지만, 위의 경우에 관계의 변화는 기존의 이진 형태의 트리를 수정해야 하는 어려움이 따른다.

3. 제안 모델

다양한 데이타 스트림과 다양한 접근 권한을 관리하는 그룹 통신에서는 안전성을 위해 다음의 항목들을 만족해야 한다.

그룹 안전성(Group security)

그룹 외부에 있는 공격자는 모든 데이타 스트림을 접근할 수 없어야 한다. 즉 모든 V_i의 키인 K_i중 하나도 공격자가 알 수 없어야 한다.

접근 제어

한 유저 서브그룹 U_i의 사용자 u는 권한이 없는 데이타 스트림을 접근할 수 없어야 한다. 즉, $V_j \not\leq V_i$인 모든 K_j중 하나도 알 수 없어야 한다.

전방위 안전성(Forward security)

한 유저 서브그룹 U_i의 사용자 u가 그룹을 탈퇴하는 경우, 탈퇴 후 u가 데이타 스트림을 접근하는 것을 막아야 한다. 이를 위해 u가 알 수 있었던 키인 K_i와 $V_j \leq V_i$인 모든 K_j를 갱신해야 한다.

후방위 안전성(Backward security)

한 유저 서브그룹 U_i에 새 사용자 u가 가입하는 경우, u가 가입하기 전의 데이타 스트림을 접근하는 것을 막아야 한다. 이를 위해 u가 가입하기 전에 키 K_i와 $V_j \leq V_i$인 모든 K_j를 갱신해야 한다.

본 논문에서 우리는 반응적인 방식으로 접근 제어를 제공하는 그룹 통신 모델을 제안하며, 이는 공개 파라미터를 이용한 키 추출 방식을 활용한다. 기존의 HAC 분야에서 공개 파라미터를 사용한 방식은 Akl과 Taylor가 처음 제안한 후 20여년간 연구되어 왔으며, 기본 개념은 다음과 같다. DAG의 두 노드 사이에 관계 $V_j \leq V_i$가 있으면 공개 파라미터 R_{ij}가 공개되며, V_i에 속한 사용자는 K_i와 R_{ij}로부터 K_j를 추출할 수 있게 된다. 하지만, V_j에 속한 사용자는 R_{ij}로부터 K_i를 추출할 수 없게 된다.

기존의 연구들은 접근 관계의 변화와 그에 따른 공개 파라미터의 갱신에 초점을 맞추고 있다. 하지만 그룹 통신의 경우 전방위 안전성과 후방위 안전성을 보장해야 하기 때문에 다른 방식의 접근 방법을 필요로 한다. 예를 들어, 기존의 사용자가 그룹을 탈퇴하는 경우 그가 속했던 노드의 키 뿐만이 아니라, 모든 후계자 노드의 키들도 새로운 키로 갱신되어야 하는데, 지난 연구들은 이를 어떤 방식으로 갱신할 것인지를 고려하지 않고 있다.

본 모델은 중앙집중식 모델이며 키를 관리하는 키 센터(Key Distribution Center: KDC)가 키를 생성/분배하는 역할을 수행한다. 이와는 별도로 누구나 접근 가능한 서버(Public parameter Access Center: PAC)를 구축해서 공개 파라미터를 기록한다. 그리고 다음의 함수들은 외부에 공개된다.

- $f: K, K \rightarrow K$ - 키 공간이 K인 단방향 함수
- $E_K(\cdot), D_K(\cdot)$ - 비밀키 K를 사용하는 암호화/복호화 암수

3.1 초기화

우선 기존의 기법들과 마찬가지로 주어진 접근 관계로부터 관계 $\leq$에 의해 부분적으로 정렬되는 DAG $G=(V,E)$를 생성한다. 그리고 각 노드 V_i에는 접근 관계에 의해 하나의 유저 서브그룹 U_i, 또는 하나의 데이타 서브그룹 D_i, 또는 하나의 유저 서브그룹 U_i와 하나의 데이타 서브그룹 D_i가 속하게 된다.

KDC는 각 유저서브그룹 U_i에 대해 임의의 서브그룹 키 SK_i를 생성한다. 그리고 LKH, OFT등의 단일 권한의 그룹키를 위한 기법을 사용해서 루트키가 SK_i인 서브그룹을 구성하게 된다. 그리고 각 사용자에게 해당하는 키를 나누어준다.

DAG의 각 노드 V_i에 대해, 만약 U_i가 V_i에 속해 있다면 $K_i = f(SK_i, null)$로 설정된다. 그리고 U_i의 각 사용자도 K_i를 추출해서 저장하게 된다. 그렇지 않은 경우에는 KDC가 임의의 키 K_i를 생성하게 된다. 각 데이타 스트림을 전송할 때에는 그에 대응되는 D_i가 속한 노드의 키인 K_i를 사용해서 암호화한 후 전송하게 된다.

그리고 KDC는 $V_j \le V_i$인 모든 i,j에 대해서 다음과 같은 공개 파라미터 R_{ij}를 계산한 후에 PAC에게 넘겨주게 된다: $R_{ij} = E_{K_i}(K_j)$.

예를 들어, 그림 1을 위한 초기화가 완료되면 U_1에 속한 사용자는 서브그룹의키 SK_1과 서브그룹 관리를 위한 키들, 그리고 SK_1으로부터 파생된 $K_i = f(SK_1, null)$을 가지게 된다. 그리고, PAC에 저장되어서 외부에 공개되는 파라미터들은 표 1과 같다. 다른 노드 V_i에 대한 키는 그림 3과 같이 K_1과 공개 파라미터를 통해서 계산하게 된다.

표 1 PAC에 저장되는 공개 파라미터 R_{ij}

	V_1	V_2	V_3	V_4	V_5	V_6
V_1	-	$E_{K_1}(K_2)$	$E_{K_1}(K_3)$	$E_{K_1}(K_4)$	$E_{K_1}(K_5)$	$E_{K_1}(K_6)$
V_2	-	-	-	$E_{K_2}(K_4)$	$E_{K_2}(K_5)$	-
V_3	-	-	-	-	$E_{K_3}(K_5)$	$E_{K_3}(K_6)$
V_4	-	-	-	-	-	-
V_5	-	-	-	-	-	-
V_6	-	-	-	-	-	-

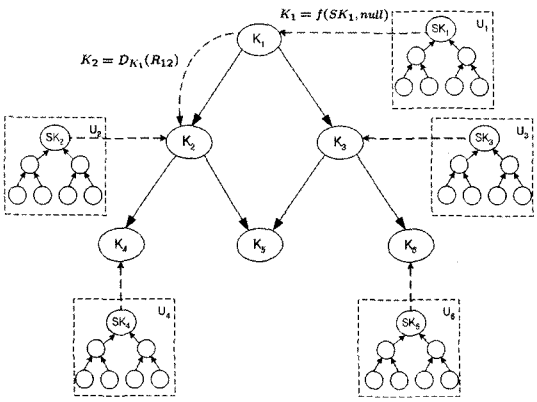

그림 3 키 분배와 추출 관계

3.2 데이타 스트림 접근

유저 서브그룹 U_i에 속한 사용자가 데이타 스트림 D_j를 접근하려 한다고 하자. 만약 $i=j$이면 이 사용자는 자신이 가진 키 K_i를 통해 복호화를 수행한다. 만약 $i \neq j$이면 이 사용자는 PAC에서 공개 파라미터 R_{ij}를 접근하게 된다. 그리고 이를 이용해서 $K_j = D_{K_i}(R_{ij})$를 추출한다. 이때, 이 사용자가 D_j에 관한 권한이 없다면 PAC에는 R_{ij}가 존재하지 않기 때문에 K_j를 알아낼 수 없게 된다.

3.3 키 갱신

새로운 사용자가 U_i에 가입하거나, 혹은 기존의 U_i의 사용자가 탈퇴함으로써 유저 서브그룹 U_i의 멤버쉽이 변하면, 우선 서브그룹 관리 기법을 통해 서브그룹키를 갱신한다. 이때 U_i의 루트키는 새로운 임의의 SK_i'로 갱신된다. 그리고 노드의 키 $K_i' = f(SK_i', null)$로 갱신된다.

다음으로, U_i의 사용자가 접근할 수 있었던 키들, 즉, $V_j \le V_i$인 모든 노드의 키 K_j가 새로운 키로 갱신되어야 한다. 만약 새로운 사용자가 그룹에 가입하는 경우라면, 이 사용자가 과거의 데이타에 접근할 수 없도록 하기 위해(후방위 안전성) 이 사용자가 U_i의 사용자로써 접근할 수 있는 모든 노드의 키가 갱신되어야 한다. 그리고, 만약 한 사용자가 그룹을 탈퇴하는 경우라면, 이 사용자가 미래의 데이타에 접근할 수 없도록 하기 위해(전방위 안전성) 이 사용자가 알 수 있었던 모든 노드의 키가 갱신되어야 한다. 즉, 가입과 탈퇴의 경우에 동일하게 해당 서브그룹의 사용자가 접근할 수 있는 모든 노드의 키가 갱신되어야 한다.

U_j가 V_j에 속해 있다면, U_j의 사용자와 KDC는 현재의 서브그룹키 SK_j를 이용해서 다음과 같이 노드 키를 $K_j' = f(SK_j, K_j)$로 갱신한다. 이렇게 함으로써 KDC가 새로운 키를 생성해서 멀티캐스트를 통해 이를 U_j의 사용자들에게 알려줄 필요 없이 각 사용자가 스스로 다음 키를 얻게 된다. 그리고 만약 U_j가 V_j에 속해있지 않은 경우에는 KDC가 새로운 키 K_j'을 선택하고 이 키는 D_j의 데이타 스트림을 암호화해서 전송할 때 사용된다.

그리고 KDC는 바뀐 키들에 대한 공개 파라미터를 변경해야 한다. 각각의 갱신된 키 K_j'에 대해서 $V_j \le V_i$인 모든 i에 대해 $R_{ij}' = E_{K_i}(K_j')$을 계산해서 이를 PAC에게 전송한다. 또한 $V_k \le V_j$인 모든 k에 대해 $R_{jk}' = E_{K_j'}(K_k)$을 계산해서 PAC에게 전송한다.

예를 들어, U_2의 사용자 u가 그룹을 탈퇴한다고 하자. 우선 KDC는 u를 제외한 다른 U_2의 사용자들에게 새로운 서브그룹키 SK_2'을 알려주게 되고, 각 사용자는 $K_2' = f(SK_2', null)$을 계산한다. 다음으로 u가 접근할 수 있었던 다른 노드의 키 K_4, K_5가 갱신된다. U_4의 사용자는 $K_4' = f(SK_4, K_4)$를 계산하고, KDC는 K_5'를 새로 선택해서 D_5에 속하는 데이타 스트림을 암호화할 때 이 키를 사용하게 된다. 그리고 KDC는 변경되는 공개 파라미터들 $R_{12}', R_{14}', R_{24}', R_{15}', R_{25}', R_{35}'$을 계산해서 PAC에 넘겨주게 된다.

3.4 접근 관계 변화

많은 애플리케이션의 경우, 기존의 유저 서브 그룹과 데이타 서브 그룹간의 접근 관계에 변화가 발생할 수 있다. 예를 들어, 한 유저 서브 그룹에 새로운 데이타 스트림에 대한 접근 권한을 추가하는 경우가 있을

수 있고, 일부 유저 서브 그룹에서만 접근할 수 있는 새로운 데이타 스트림이 추가되는 경우도 있을 수 있다. 이러한 경우에 DAG G는 변화된 접근 관계를 고려한 DAG $G'=(V',E')$으로 변경되며, 이를 반영하기 위해서 기존의 G에서 일련의 노드를 추가/제거하는 과정과 에지를 추가/제거하는 과정들을 다음과 같이 수행한다.

3.4.1 에지 추가

$V_k \leq V_i$가 되도록 두 노드 V_i, V_k사이에 새로운 에지를 추가하는 경우를 생각해보자. A_i를 에지가 추가되기 전에 U_i의 사용자가 알 수 있는 키의 집합이라 하고 $(A_i=\{K_j|V_j \leq V_i\})$, A_i'을 에지가 추가된 후 알 수 있는 키의 집합이라 하자. 에지가 추가된 뒤의 후방위 안전성을 위해서는 $K_j \in A_i' - A_i$인 모든 키들이 갱신되어야 한다. 각각의 K_j에 대해서, V_j에 U_j가 속한 경우에 각 U_j의 사용자는 $K_j'=f(SK_j,K_j)$로 갱신하게 되고, 그렇지 않은 경우 KDC가 새로운 키 K_j'을 생성한다. 그리고 KDC는 갱신된 키 K_j'에 대해서 $V_j \leq V_i$인 모든 i에 대해 $R_{ij}'=E_{K_i}(K_j')$을 계산하고 $V_k \leq V_j$인 모든 k에 대해 $R_{jk}'=E_{K_j'}(K_k)$을 계산해서 이를 PAC에게 전송한다.

3.4.2 에지 제거

두 노드 V_i, V_k사이의 에지를 제거하는 경우도 비슷하다. A_i는 에지가 제거되기 전에 U_i의 사용자가 알 수 있는 키의 집합이라 하고 A_i'은 에지가 제거된 후의 키 집합이라 하면, 이 경우 전방위 안전성을 위해 갱신되어야 하는 키들은 $K_j \in A_i - A_i'$이고 이 키들은 에지 추가와 같은 방법으로 갱신된다.

3.4.3 노드 추가

새로 추가되는 노드 V_i에 새로운 유저 서브그룹 U_i가 속하게 되는 경우, KDC는 새 서브그룹키 SK_i가 루트키가 되도록 서브그룹을 구성한다. V_i에 새로운 데이타 서브그룹 D_i만이 속하는 경우, KDC가 새 서브그룹키 SK_i를 임의로 생성하게 된다.

노드가 추가된 후에는, 새로 추가된 노드 V_i와 기존의 노드 V_j 사이에 새로운 에지가 추가될 경우 이에 대한 과정을 수행한다.

3.4.4 노드 제거

기존의 노드 V_i가 제거되는 경우, 우선 V_i와 다른 노드 V_j 사이의 모든 에지가 제거 과정을 통해 제거된다. 그 후, KDC는 PAC에 기록되어 있는 공개 파라미터 중 V_i와 관계된 모든 $R_{ij}, R_{ji}(V_j \in G')$를 삭제함으로써 V_i를 제거한다.

4. 성능 분석

기존의 방식들은 키 갱신이 발생할 때마다 매번 모든 사용자가 자신의 키를 갱신해야 하는 선행적 방식이었으며, 이에 대한 대안으로 본 논문에서는 자신이 접근하려는 스트림에 대해서만 해당하는 키를 알아내는 반응적 방식의 키 관리 기법을 제안하였다. 이번 장에서는 제안된 방식의 성능을 분석한 결과를 보여준다. 이때, 서브그룹을 관리하는 방법은 잘 알려진 논리적 키 트리를 사용하며, 자세한 과정은 생략하기로 한다.

우선 사용자의 입장에서 필요한 오버헤드를 분석하고 다음으로 KDC에서 필요한 오버헤드를 분석한다. 그리고 선행적 방식과 비교했을 때 반응적 방식의 장점을 설명한다.

4.1 사용자 오버헤드

4.1.1 키 저장량

표 2는 HAC 키 그래프와 제안된 기법에서 한 사용자 서브그룹 U_i에 속한 한 사용자 u가 저장해야 하는 키들을 비교하고 있다. 이 때, $|U_i|$는 서브그룹에 속한 사용자의 수를 의미한다. $A_i=\{K_j|V_j \leq V_i\}$는 u가 접근할 수 있는 모든 노드의 키 집합을 의미하고, $|A_i|$는 A_i의 원소의 수를 나타낸다. 그리고, 각 사용자는 c개의 최근 키를 캐쉬에 저장한다고 가정하며, 일반적으로 $c<<|A_i|$이다. 표에서 알 수 있듯이, HAC 키 그래프에서는 접근 제어를 위해 최대 G의 노드 개수만큼의 키들을 저장해야 하지만 제안된 기법에서는 자신이 속한 노드의 키 하나와 캐쉬된 키들만을 저장하게 된다.

표 2 $u \in U_i$에 의해 저장되는 키 개수의 비교

	HAC 키 그래프	제안 모델
서브그룹키	SK_i	SK_i
서브그룹 관리를 위한 키 개수	$\log\|U_i\|$	$\log\|U_i\|$
접근 제어를 위한 키 개수	$\|A_i\|$	$1+c$

4.1.2 데이타 스트림 접근

U_i의 사용자 u가 한 데이타 스트림 d를 접근하려 한다고 하자. 이 경우는 다음과 같이 두 가지로 나뉠 수 있다.

- $d \in D_i$: 이 경우 u가 소유한 K_i를 통해 바로 접근
- $d \in D_j$ $(V_j \leq V_i)$: 이 경우 u는 PAC로부터 R_{ij}를 얻어와서 K_j를 계산

두 번째의 경우, 한번 K_j를 알아낸 후에 이를 캐쉬에 저장해 놓으면 다음번에 K_j가 필요할 때 매번 R_{ij}를 얻어올 필요 없이 K_j가 갱신되었을 때에만 얻어오면 된다.

표 3 키 갱신을 위해 u에 의해 수행되는 코스트 비교

	HAC 키 그래프		제안 모델	
	전송된 메시지 크기	계산량	전송된 메시지 크기	계산량
$u \in U_i$	$(\log\|U_i\|+\|A_i\|)$L	$(\log\|U_i\|+\|A_i\|)$D	$(\log\|U_i\|)$L	1f
$u \in U_j (V_j \leq V_i)$	$(\log\|U_i\|+\|A_i\|)$L	$(\log\|U_i\|+\|A_j\|)$D	$(\log\|U_i\|)$L	$(\log\|U_i\|)$D
$u \in U_j (V_j \not\leq V_i)$	$(\log\|U_i\|+\|A_i\|)$L	$(\log\|U_i\|+\|A_i \bigcap A_j\|)$D	$(\log\|U_i\|)$L	-

L: 암호화된 키의 크기, D: 복호화 연산량, f: 단방향 함수 연산량

또한 d를 접근하는 중에 K_j가 갱신된다면 그때에만 새로운 R_{ij}를 얻어오게 된다. 즉, K_j를 위해 R_{ij}를 PAC로부터 얻어오는 횟수는 K_j가 갱신되는 횟수보다 작거나 같다.

4.1.3 멤버쉽 변화를 위한 키 갱신

U_i에 새 사용자가 가입하거나 한 사용자가 탈퇴함으로써 멤버쉽이 변한다고 하자. 표 3은 이 경우에 HAC 키 그래프와 제안된 기법을 비교한 결과를 보여준다. 어떤 키가 새로 갱신되는지를 알려주는 메시지는 양쪽 기법에서 모두 제외하였다. HAC 키 그래프에서 실제로는 이진 트리를 확장한 것이기 때문에 이는 일부의 경우에 전송되는 메시지의 크기를 조금 감소시키는 효과를 가지지만 이 경우 저장되는 키의 개수와 연산량이 늘어나는 반대 효과를 가진다. 이번 비교에서는 이러한 세부사항을 고려하지 않고 전반적인 비교를 수행하였다.

표에서 알 수 있듯이 제안된 기법은 각 사용자가 전송받는 메시지의 양을 크게 줄이고 마찬가지로 키 갱신을 위한 연산량도 준다는 장점을 가진다. 이는 제안된 기법이 반응적 방식을 따르기 때문으로, 반대로 위에서 설명한 데이타 스트림의 접근을 위한 오버헤드가 추가된다. 반응적 방식과 선행적 방식에 대한 비교는 4.3에서 자세히 설명한다.

4.1.4 접근 관계 변화를 위한 키 갱신

변화된 접근 관계를 반영하기 위해 DAG G가 변하는 경우, 이는 일련의 에지 추가/제거와 노드 추가/제거를 통해 이루어진다. 이때, 에지의 추가/제거는 일부의 사용자가 키들을 갱신함으로써 반영되며, 노드의 추가/제거는 위해 각 사용자가 수행해야 할 오버헤드는 없다. 표 4는 V_i에서 V_k로의 에지가 추가/제거되는 경우 키 갱신을 수행해야 하는 사용자와 그때의 연산량을 나타낸다.

반면에, 기존의 HAC 키 그래프에서는 접근 관계의 적은 변화도 반영하기 힘들다. 이는 이 그래프가 이진 트리를 기반으로 생성된 것이기 때문이며, 새로운 접근 관계를 위해 다시 그래프를 생성하는 것 외에 간단한 해결방안이 제시되지 않았다. 그리고 이는 각 사용자에게 다시 키들을 분배해야 하는 것으로 매우 큰 오버헤드를 요구한다.

표 4 접근 관계 변화를 위한 키 갱신

	갱신에 참여하는 사용자수	연산량
에지 추가	$u \in V_j$ where $K_{j \in A_i' - A_i}$	1f
에지 제거	$u \in V_j$ where $K_{j \in A_i - A_i'}$	1f

4.2 KDC 오버헤드

이제 KDC가 수행해야 하는 비용을 생각해 보자. 초기화 과정을 제외하면 KDC가 수행해야 하는 역할은 키들을 갱신하고 해당하는 공개 파라미터를 계산해서 PAC에게 전송하는 것이다. 키 갱신 중 하나의 서브그룹키 갱신을 위한 과정은 기존의 기법들과 동일한 비용을 필요로 하므로 분석에서 제외하도록 한다.

KDC가 한 노드 V_i의 키 K_i를 갱신하는 경우는 멤버쉽이 변화하는 경우와 접근 관계 변화로 인한 에지 추가/제거를 수행하는 경우이다. KDC는 우선 단방향 함수 f를 이용하거나 혹은 새로운 키를 선택해서 K_i'으로 갱신한 후에, 다음의 공개 파라미터들을 계산한다.

- $R_{ji}' = E_{K_j}(K_i')$ for all j where $V_i \leq V_j$
- $R_{ij}' = E_{K_i'}(K_j)$ for all j where $V_j \leq V_i$

즉, 하나의 키가 갱신되면 최대 $|V|$-1개의 공개 파라미터가 계산되어서 PAC에 전송된다.

많은 경우에 KDC는 여러 개의 키를 한꺼번에 갱신하게 되며, 이러한 키의 개수는 높은 접근 권한을 가지는 노드의 멤버쉽이 변할수록 증가하게 된다. 최악의 경우에 완전히 정렬된 접근 관계를 가지는 상황에서 가장 권한이 높은 노드의 키가 갱신된다면, 이는 모든 노드의 키가 갱신되어야 하며 모든 공개 파라미터도 새로 계산되어야 한다. 이 경우 계산되어서 전송되는 공개 파라미터는 최대 $\frac{(|V|-1)(|V|-2)}{2}$이며 계층이 매우 크다면 KDC에게 부담이 될 수 있다. 하지만, 현실적으로 |V|의 크기는 수십에서 수백 개 사이이며 이에 대한 계산은 현재 서버의 사양으로 봤을 때 최대 수초의 연산만을 요구한다[14]. 그리고 이 파라미터들은 네트워크 전체에 멀티캐스팅 되는 것이 아니라 PAC에게만 전송되는 것으로 전체 네트워크의 성능에 큰 영향을 주지 않는다.

만약 |V|의 크기가 매우 커서 공개 파라미터 계산 및 전송이 힘든 경우, 다음의 대안을 사용할 수 있다. 원래 방법에서는 모든 $V_j \leq V_i$에 대해 R_{ij}를 유지했지만, 이 대신에 $V_j \leq V_i$이고 두 노드가 직접적으로 연결되어 있는 경우에만 R_{ij}를 유지한다. 그리고 $V_j \leq V_k \leq V_i$이고 U_i의 사용자가 D_j를 접근하려 하는 경우에, 이 사용자는 PAC로부터 R_{ik}, R_{kj}를 얻어와서 이를 통해 K_j를 얻게 된다. 이 경우 키 갱신시 바뀌는 공개 파라미터는 최악의 경우에도 DAG의 에지의 수(|E|)만큼이고, DAG는 부분적으로 정렬된 집합을 반영한 것이기 때문에 에지의 개수는 n보다 그리 않다. 이 대안의 경우에는 사용자의 오버헤드가 늘어난다는 단점이 생기므로 주의깊게 두 비용을 비교해서 적합한 방식을 선택해야 한다.

4.3 선행적 방식과 반응적 방식 비교

기존의 선행적 방식과 제안된 반응적 방식에서 사용자가 키를 바꾸는 주기를 생각해 보자. 선행적 방식에서 일반적으로 한 멤버의 멤버쉽 변화가 t 주기마다 발생하고 전체 사용자가 n이라면, 평균적으로 $\frac{t}{n}$ 주기마다 키 갱신을 수행한다. 만약 n이 매우 큰 그룹이라면 키 갱신도 매우 빈번하게 발생함을 알 수 있다. 예를 들어 한 멤버의 그룹 가입/탈퇴가 1년마다 발생하고 전체 멤버의 수가 1,000,000명인 경우, 키 갱신은 대략 30초에 한 번 꼴로 발생하게 된다. 하지만 반응적 방식에서는 사용자가 자신이 접근하는 데이타 스트림을 변경하는 주기가 t'이라 할 때, 이는 그룹 크기에 상관이 없고 사용자의 취향과 애플리케이션의 특성에 달려있다. 또한 각 사용자는 자신이 접근할 수 있는 모든 데이타 스트림을 접근하는 것이 아니고 취향에 따라 일부 데이타 스트림에 편중되는 경향이 있다. 그러므로, 제안된 방식은 그룹 크기가 매우 크고 사용자가 접근하는 스트림이 빈번하게 바뀌지 않는 형태의 모델일수록 선행적 방식보다 더 좋은 성능을 보임을 알 수 있다.

선행적 방식의 한계 중 하나는 사용자가 모든 키 갱신 메시지를 놓치지 않고 전송받아야 한다는 점이다. 갱신 메시지 중 하나를 패킷 로스 등으로 인해 놓치게 된다면 그 뒤에 키들을 알아낼 수 없게 되고 다른 사용자와 동기화되지 못하며, 결국 그룹을 탈퇴하고 재가입해야 한다. 반면에 반응적 방식에서는 항상 접근 가능한 PAC를 통해 과거 키 갱신 과정을 알 필요가 없이 현재의 키를 얻을 수 있다. 물론 제안된 모델도 자신의 서브그룹의 멤버쉽 변화로 인한 키 갱신 메시지는 손실 없이 전송받아야 한다. 하지만, 이는 전체 그룹의 모든 키 갱신 메시지를 손실 없이 받아야 하는 것보다 훨씬 안정성이 높다.

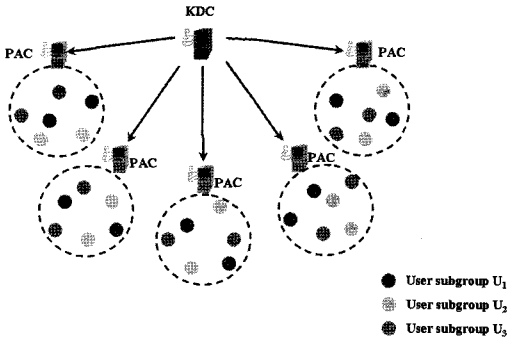

그림 4 부하를 분산시키기 위한 PAC의 중복 배치

PAC에 한번에 많은 사용자가 접근해서 과부하가 걸리는 큰 규모의 그룹인 경우, 성능 향상을 위해 여러 PAC를 중복되게 설정할 수 있다. 즉, 그림 4와 같이 네트워크의 곳곳에 PAC를 설치하고 각 사용자는 가까운 곳에 위치한 PAC를 접근함으로써 부하를 분산시킬 수 있다. 이는 PAC가 공개되는 값만을 가지기 때문에 가능한 것으로, 이로 인해 안전성이 저하되지 않는다.

5. 안전성 분석

이번 장에서는 제안된 모델이 앞서 설명한 안전성을 위한 요구사항을 만족함을 보인다.

그룹 안전성

모든 데이타 스트림은 자신이 속한 D_i에 해당하는 키 K_i를 통해 암호화되서 전송된다. 그리고 외부의 공격자가 알 수 있는 정보는 이런 암호문과 PAC에 기록된 공개 파라미터들이다. 하지만, 이 공개 파라미터도 키 K_j를 키 K_i로 암호화한 형태의 암호문으로 공격자는 이 두 키를 모두 모르는 상태이다. 그러므로 공격자가 할 수 있는 가능한 공격은 모든 가능한 키를 다 검색하는 것뿐이다.

접근 제어

유저 서브그룹 U_i에 속한 공격자가 $V_j \not\leq V_i$인 j에 대해 키 K_j를 알아내려 한다고 하자. 이 경우, K_i와 K_j는 상호 독립적이므로 K_i로부터 K_j를 알아낼 수 없다. 공개 파라미터를 이용한 가능한 공격 방법은 $V_k \leq V_i, V_k \leq V_j$인 k에 대해 K_k과 $R_{jk} = E_{K_j}(K_k)$ 쌍을 이용하는 알려진 평문을 이용한 공격(known plaintext attack)이다. 하지만, 일반적으로 이런 형태의 공격은 매우 많은 쌍을 필요로 하며 현재 사용중인 AES등의 암호 알고리즘은 이러한 형태의 공격에 대해 현실적으로 안전하다.

전방위 안전성

U_i의 사용자 u가 탈퇴하는 경우, 서브그룹키 관리 기법에 의해 u는 새 서브그룹키 SK_i'을 알 수 없게 된다. 그리고 노드 키는 $K_i' = f(SK_i', null)$로 갱신되므로 마찬가지로 u가 알 수 없다. 다음으로 $V_j \leq V_i$인 모든 K_j는 오직 U_j의 서브그룹키를 알고 있는 서브그룹 멤버만이 계산할 수 있는 K_j'으로 갱신되므로 u가 알 수 없다. 그리고 공개 파라미터들 역시 갱신된 키들을 이용해서 변경되므로 과거의 키와 공개 파라미터를 사용해서 현재의 키를 알아낼 수 없다.

후방위 안전성

U_i에 새 사용자 u가 가입하는 경우, K_i'과 K_i는 상호 독립적이므로 u가 K_i'으로부터 K_i를 알아낼 수 없다. 그리고 $V_j \leq V_i$인 $K_j' = f(SK_j, K_j)$의 경우, 함수 f의 단방향성으로 인해 SK_j와 K_j를 모두 알 수 없다. 과거의 키들 K_i와 모든 K_j를 알지 못하므로 현재의 키와 과거의 공개 파라미터를 사용해서 과거의 키를 알아낼 수 없다.

6. 결 론

본 논문에서는 그룹 통신에서의 접근 제어를 위한 기존의 선행적 방식의 대안으로 반응적 방식의 기법을 제안하였다. 제안된 기법에서 각 사용자는 자신이 접근 가능한 모든 키를 관리할 필요 없이 최소한의 키만을 관리하며, 다른 키는 PAC에 있는 공개 파라미터를 통해 접근할 수 있게 한다. 이를 통해 사용자가 고려해야 하는 키 갱신 과정이 간소화되었으며, 복잡한 접근 관계를 가지는 큰 규모의 그룹에 대해 성능의 향상을 얻을 수 있다. 또한, 제안된 방식은 접근 관계가 능동적으로 변할 때에도 쉽게 이를 반영한다는 장점을 가진다.

참 고 문 헌

[1] M. Valdvogel, G. Caronni, D. Sun, N. Weiler, B. Plattner, "The versakey framework: versatile group key management," IEEE JSAC special issue on Service Enabling Platforms For Networked Multimedia Systems, Vol.17, No.9, 1999.

[2] M. Burmester, Y. Desmedt, "A secure and efficient conference key distribution system," In advances in cryptology, Eurocrypt '94, pp. 275-286, 1994.

[3] M. Steiner, G. Tsudik, M. Waidner, "Key agreement in dynamic peer groups," IEEE transactions on Parallel Distributed Systems, Vol.11, No.8, pp. 769-780, 2000.

[4] Y. Kim, A. Perrig, G. Tsudik, "Tree-based group key agreement," ACM transactions on Information and System Security, Vol.7, No.1, pp. 60-96, 2004.

[5] C. K. Wong, M. Gouda, S. Lam, "Secure group communications using key graphs," In Proceedings of the ACM SIGCOMM '98, pp. 68-79, 1998.

[6] J-C. Birget, X. Zou, G. Noubir, B. Ramamurthy, "Hierarchical-based access control in distributed environments," In Proceedings of ICC 2001, 2001.

[7] S. G. Akl, P. D. Taylor, "Cryptographic solution to a problem of access control in a hierarchy," ACM transactions on Computer Systems, Vol.1, No.3, pp. 239-247, 1983.

[8] C. H. Lin, "Dynamic key management schemes for access control in a hierarchy," Computer Communications, Vol.20, pp. 1381-1385, 1997.

[9] C. Yang, C. Li, "Access control in a hierarchy using one-way hash functions," Computers & Security, Vol.23, pp. 659-664, 2004.

[10] X. Zou, "Secure group communications and hierarchical access control," PhD. Thesis, University of Nebraska-Lincoln, USA, 2000.

[11] Q. Zhang, Y. Wang, "A centralized key management scheme for hierarchical access control," In Proceedings of IEEE Globecom, pp. 2067-2071, 2004.

[12] Y. Sun, K. J. Ray Liu, "Scalable hierarchical access control in secure group communications," In Proceedings of IEEE INFOCOM, pp. 1296-1306, 2004.

[13] Y. Kim, A. Perrig, G. Tsudik, "Communication-efficient group key agreement," In Proceedings of the 16th international conference on Information security: Trusted information, pp. 229-244, 2001.

[14] http://www.cpktec.com/performance.html

김 희 열

2000년 한국과학기술원 전산학과 학사 2002년 한국과학기술원 전자전산학과 석사. 2006년 한국과학기술원 전자전산학과 박사. 2007년 삼성전자 책임연구원. 관심분야는 컴퓨터 보안, 무선 네트워크 보안, 응용 암호

이 윤 호

2000년 한국과학기술원 전산학과 학사 2002년 한국과학기술원 전자전산학과 석사. 2006년 한국과학기술원 전자전산학과 박사. 2007년 한국과학기술원 연수연구원. 관심분야는 컴퓨터 보안, 무선 네트워크 보안, 응용 암호

박 용 수

1996년 KAIST 전산학과(학사). 1998년 서울대학교 컴퓨터공학과(석사). 2003년 서울대학교 전기컴퓨터공학부(박사). 2003년~2004년 서울대학교 자동제어특화연구센터 박사후연수연구원. 2005년~현재 한양대학교 정보통신대학 컴퓨터전공 조교수

윤 현 수

2007년 1월~현재 한국대학정보화협의회 부회장. 2007년 1월~2008년 1월 한국정보과학회 부회장. 2006년~현재 한국공학한림원 회원. 1989년 7월~현재 KAIST 조교수, 부교수, 정교수, 영년직 교수. 2005년 6월~현재 KAIST 학술정보처장, 정보시스템 연구소장. 2000년 8월~2004년 2월 KAIST 정보보호 학제전공 책임교수. 2005년 1월~현재 한국정보처리학회 이사. 2001년 3월~2003년 2월 한국정보과학회 이사 2000년 3월~2005년 1월 한국정보처리학회 논문지 편집위원장. 1988년 8월~1989년 8월 AT&T Bell Labs MTS. 1979년 1월~1984년 6월 삼성전자 연구원