情報保護學會論文誌
第13券 第5號, 2003. 10

Diffie-Hallman 키 교환을 이용한 확장성을 가진 계층적 그룹키 설정 프로토콜

박 영 희*, 정 병 천*, 이 윤 호*, 김 희 열*, 이 재 원*, 윤 현 수*

Scalable Hierarchical Group Key Establishment using Diffie-Hellman Key Exchange

Younghee Park*, Byungchun Chung*, Youn-Ho Lee*, Heeyoul Kim*,
Jaewon Lee*, Hyoonsoo Yoon*

요 약

안전한 그룹 통신은 그룹에 속한 멤버들만이 안전하면서도 비밀스럽게 서비스를 받을 수 있도록 보장하는 것이며, 안전한 그룹 통신을 하기 위해서는 안전하게 그룹키를 분배하고, 그룹 멤버들에 대한 효율적인 관리가 중요하다. 또한, 그룹키를 생성, 분배 그리고 갱신하는데 있어 적은 통신량과 연산량이 필요하다. 본 논문에서는 그룹키를 생성하기 위해서, 잘 알려진 두 멤버간의 Diffie-Hellman 키 교환과 완전 이진 트리를 이용한 새로운 방법을 제안한다. 본 논문에서는 많은 양의 모듈러 멱승 연산의 일부를 곱셈 연산으로 대체하여 한 멤버에게 필요한 멱승 연산량을 최소화 시킨다. 그리하여, 제안한 프로토콜은 그룹의 크기에 관계없이 프로토콜을 수행하기 위하여 각 그룹의 멤버는 항상 일정한 모듈러 멱승 연산과 곱셈 연산을 수행하며, 기존 프로토콜보다 전체 모듈러 멱승 연산량도 적다.

ABSTRACT

The secure group communication enables the members, which belong to the same group, to communicate each other in a secure and secret manner. To do so, it is the most important that a group key is securely distributed among them and also group membership is efficiently managed. In detail, the generation, the distribution and the refreshment of a group key would be highly regarded in terms of low communication and computation complexity. In this paper, we show you a new protocol to generate a group key which will be safely shared within a group, utilizing the 2-party Diffie-Hellman key exchange protocol and the complete binary tree. Our protocol has less complexity of computation per group member by substituting many parts of exponentiation computations for multiplications. Consequently, each group member needs constant computations of exponentiation and multiplication regardless of the group size in the protocol and then it has less complexity of the computation than that of any other protocols.

keyword : *secure group communication, group key, Diffie-Hellman key exchange*

Ⅰ. 서 론

안전한 그룹 통신이란, 어떤 목적아래 모인 특정 그룹이 있을 때, 그 그룹에 속한 멤버들만이 그룹의 정보를 송수신할 수 있도록 비밀스런 통신을 보장하는 것이다. 즉, 어떤 몇몇 멤버들이 정보를 공유하고

* 한국과학기술원 전자전산학과 전산학전공 컴퓨터구조연구실(jenifer02@mail.kaist.ac.kr)

자 형성된 한 그룹이 있을 때, 이때 형성된 그룹의 멤버만이 그룹의 비밀 정보를 얻을 수 있어야 하며, 그룹에 속하지 않은 사람이나 도청자와 같은 공격자들은 그룹의 비밀 정보를 얻을 수 없어야 한다. 이와 같이, 그룹 멤버들만이 정보를 효율적이면서 안전하게 공유하기 위해서는 그룹키를 이용한 암호 통신이 필요하다. 이 그룹키는 그룹에 속한 멤버들만이 알고 있어야 하며, 그룹 멤버들의 변화에 따라 그룹키도 갱신되어야 한다.

안전한 그룹 통신을 위해서는 그룹키 설정과 그룹 멤버 관리가 필요하다. 다시 말해, 특정 그룹에 속해 있는 멤버들만이 그룹키를 생성할 수 있고 생성된 그룹키는 그룹 멤버들만 공유해야 한다. 그리고, 여러 가지 원인으로 그룹멤버들에게 변화(가입, 탈퇴, 병합, 분할)가 발생하였을 때, 그룹멤버들은 적절한 연산을 통하여 그룹키를 갱신하고 갱신된 그룹키를 현재 남아있는 멤버들에게만 재분배하는 그룹 멤버 관리가 필요하다.

키를 설정하는데 있어서, 신뢰하는 서버들이 그룹키를 생성하는 방법과 모든 그룹 구성원들이 참여하여 그룹키를 생성하는 방법이 있다. 첫 번째 방법은 TTP(Trusted Third Party)와 같은 신뢰할만한 서버나 하나 이상의 서버 역할을 하는 멤버들이 그룹키를 생성하여 안전한 채널(Secure Channel)을 통해 나누어 주는 방법이다. 이는 서버가 통신에 관련된 모든 키들을 연산하고 안전하게 키를 공유하며, 서버가 그룹 멤버들을 관리한다. 이 방식들은 서버가 모든 필요한 연산을 수행함으로 빠르게 키를 생성할 수 있지만, 항상 신뢰할만한 서버는 외부 공격자들의 공격대상이 될 수 있다. 또한, 서버에 문제가 발생하면 전체 그룹 통신이 마비되는 단일 장애점(Single Point of Failure) 문제가 발생한다. 무엇보다도 서버를 이용하는 방법들은 그룹멤버들사이에 존재하는 많은 안전한 채널을 관리해야 하며, 안전한 채널을 유지하고 관리하는 비용이 많이 든다.[1,4,9,11] 두 번째 방법은 그룹을 구성하는 모든 멤버들이 그룹키를 생성하는데 똑같이 참여하는 방법이다.[1,2,12,13,10] 이 방법에 의해 생성된 그룹키는 어떤 특정 서버가 미리 연산한 값이 아니라, 그룹멤버들 각자가 비밀스럽게 생성한 각 그룹멤버의 비밀값으로 이루어진다. 그리하여, 모든 그룹멤버들이 참여하기 전까지 생성될 그룹키를 미리 예측할 수 없으며, 매번 생성되는 키는 이전 키와 다른 최신(fresh)의 그룹키를 만들 수 있다. 그러나, 이런 방법들은 그룹 멤버들 각자가 그룹키를 연산하기 때문에 그룹 구성원들에게 많은 모듈러 멱승 연산량에 대해 부담을 줄 뿐만 아니라, 그룹이 크면 클수록 한 그룹멤버가 부담해야하는 연산량 또한 크다. 그리고 앞서 살펴본 여러 가지 방법들은 그룹멤버에 변화가 발생하면, 그룹의 비밀 정보와 그룹키의 안전성을 보장하기 위해서 그룹키를 갱신해야 한다. 서버가 존재하는 방법은 서버가 그룹 멤버들에 대한 관리를 전적으로 맡게 되지만, 서버없이 모든 그룹 멤버들이 그룹키를 생성하는 방법들은 그룹 구성원이 모두 협력하여 적절한 연산을 통해 그룹 멤버들을 관리하고 그룹키도 갱신한다. 그 뿐만 아니라, 변화가 일어나는 위치와 그룹의 크기에 따라 한 그룹 멤버에게 주어진 연산량이 부담이 된다.

본 논문에서는 두 멤버간의 Diffie-Hellman 키 교환 프로토콜과 완전 이진 트리(Complete Binary Tree)를 이용하여 서버없이 모든 그룹의 멤버들이 참여하여 그룹키를 생성하는 새로운 방법을 제안한다. 완전 이진 트리의 계층적 구조에 기반함으로 통신량은 이전 연구들과 비슷하다. 그러나, 이전의 여러 프로토콜들은 그룹 크기에 따라 한 멤버에게 주어진 연산량이 변하게 되는 반면, 제안한 프로토콜은 그룹의 크기에 상관없이 그룹 구성원들 각자는 항상 일정하게 상수만큼의 모듈러 멱승 연산과 곱셈 연산을 가진다.

본 논문의 구성은 다음과 같다. 2장에서는 관련 연구에 대해 간단하게 서술하고 특히, 서버 없이 모든 그룹멤버들이 참여하여 그룹키를 생성하는 기존의 연구들에 대해 소개한다. 3장에서는 본 논문에서 제안한 새로운 방법에 대해 설명하고, 4장에서는 그룹의 변화에 따른 그룹멤버들의 관리에 대해 다룬다. 5장에서는 이 프로토콜이 요구하는 암호적 속성에 대해 논의하고, 6장에서는 기존 프로토콜들과 비교 분석한 다음 결론을 맺는다.

II. 관련 연구

안전한 그룹 통신을 위한 여러 가지 키 설정 프로토콜(Key Establishment Protocol)들이 있다. 신뢰할만한 서버가 존재하여 서버의 도움으로 그룹키를 생성하고 공유하는 방법과 서버없이 모든 구성원이 함께 참여하여 그룹키를 생성하고 공유하는 방법이 있다. [4,7]은 서버들에 의해서 그룹키를 만들고, [2,10]은 그룹을 구성하는 멤버들이 함께 참여하여 그룹키를 만든다. 이것은 서버들이 키를 생성하는 것보다

라운수가 많이 들지만, 신뢰할만한 서버들이 없으므로 단일 장애점 문제가 없다.

[9,11,12,13]은 1998년 Wong외 여러 사람에 의해 제안된 키 그리프(Key Graph)[9]라는 개념에서 시작되어, 2000년대에 키 트리(Key Tree)[12,13]라는 개념으로 발전하였다. 키 그래프란 트리에서 단말 노드(Terminal Node)들은 그룹을 구성하는 그룹멤버들로 구성되어 있고, 내부 노드(Internal Node)들은 키값들로 구성되어 있다. 즉, 이 방법은 모든 그룹멤버들이 공유하는 근 노드(Root Node)에 위치한 그룹키가 하나 있고, 내부 노드에는 그룹키를 만들기 위한 서브키(Subkey)값들로 구성되어 있다. 키 트리를 이용하면, 적은 비용으로 키를 생성할 수 있고, 특정 멤버만이 알고 있는 서브키를 이용하여 선택적이고 효율적으로 그룹키를 전달할 수 있다. 그러나 모든 그룹의 구성원들이 *logn*개의 키를 가지고 있어야 하며, 그룹키를 갱신할 때 그에 해당하는 서브키들을 모두 갱신해야 한다.

서버 없이 모든 그룹의 구성원들이 참여하여 그룹키를 만드는 방법들에 대해 간단히 살펴보면, GDH (Generic Diffie-Hellman) 프로토콜[10]은 모두 세 가지 버전이 있으며, 브로드캐스트(Broadcast)방법을 이용하지 않는 GDH.1, GDH.3과 브로드캐스트를 이용한 GDH.2가 있다. GDH.2은 여러 가지 면에서 개선된 논문들이 계속해 나오고 있고,[3,6,14] 1998년에 제안된 CLIQUES[15]은 GDH.2을 발전시킨 것이다. Becker와 Wille에 의해서 제안된 Hypercube/Octopus 프로토콜[2]에서 Hypercube 프로토콜은 작은 라운드 수를 요구하고, Octopus 프로토콜은 최소한의 메시지 수를 요구하는 프로토콜이지만, 두 프로토콜은 특정한 네트워크 구조에 의존하는 단점이 있어 그룹멤버 관리에 부적합하다. Burmester와 Desmedt은 제안한 BD 프로토콜[1]은 네가지 회의 키 분배 시스템(Conference Key Distribution System)을 제안했다. 그 중에서 본 논문에서 관심있는 것은 이진트리에 기반한 것과 브로드캐스트를 이용한 방법이다. 이진 트리에 기반한 시스템(Tree-based System)은 근 노드가 신뢰할만한 서버라고 가정한 서버가 있는 방식이며, 근 노드가 그룹에 관련한 모든 일을 하게 된다. 브로드캐스트를 이용한 BD 프로토콜은 특정한 멤버들의 구성을 요구하지 않고, 모든 멤버가 그룹키를 만들기 위해서 그룹 멤버 각자가 브로드캐스된 메시지를 얻어서 그룹 멤버들 각자가 연산을 하게 된다. 마지막으로, TGDH (Tree Group Diffie-Hellman) 프로토콜[12]과 STR 프로토콜[13]은 키 트리(key tree)와 브로드캐스트를 이용한 또 다른 방법이며, TGDH 프로토콜은 이진 트리를 이용하고 있고, STR 프로토콜은 경사 트리(Skewed Tree)를 이용하며, 각각의 그룹멤버들은 브로드캐스트 되는 BK(Blinded key)로부터 정보를 얻어 그룹키를 생성하는 새로운 방법이다. 그러나, 이 프로토콜들은 브로드캐스트되는 BK 메시지에서 그룹멤버 자신에서부터 근 노드까지 이르는 서브키(거의$O(log_2n)$들과 그룹키를 연산해야 함으로 각각의 그룹 멤버에게 연산에 대한 부담을 준다. 그룹에 변화가 발생하게 되면 후원자(Sponsor)가 이 BK를 갱신하여 브로드캐스트함으로써 그룹에 남아있는 멤버들은 키를 효율적으로 갱신할 수는 있지만, 그룹의 크기와 변화가 일어나는 트리상의 위치에 따라 그 연산량도 크게 다르다.

Ⅲ. 제안한 그룹키 설정 방법

3.1 표기 및 정의들

본 논문에서 사용되는 표기들은 [표 1]과 같다. 몇 가지만 간략하게 설명하면, 그룹키는 $F(X_i)$와 $f(x_i)$의 두 함수에 의해서 생성된 값이다. 다시 말해, 그룹키는 $F(X_i)$의 곱들이고, $F(X_i)$는 $f(x_i)$곱이 된다. 함수 $f(x_i)$는 두 멤버간의 Diffie-Hellman 키 교환으로 얻어지는 비밀값이다. 즉, 어떤 두 멤버 m_i, m_{i+1}이 있을 때, Zq 상에서 각 멤버들은 랜덤하게 비밀값 r_i, r_{i+1}을 생성한다. 그런 다음, 각 멤버는 $g^{r_i} \bmod p$와 $g^{r_{i+1}} \bmod p$를 연산하여 서로 값을 교환하게 되고, 두 멤버 m_i, m_{i+1}은 자신만이 알고 있는 랜덤한 비밀값을 멱승함으로써 같은 함수 값 $f(x)=g^{r_i r_{i+1}} \bmod p$를 연산한

〔표 1〕 표기법들(Notations)

n	전체 그룹의 멤버들의 수
l	트리의 레벨($l \in [1, log2n]$)
R,L	오른쪽 자식 노드와 왼쪽 자식 노드
i	트리상의 멤버들의 위치
mi	i-th의 그룹 멤버($i \in [1, n]$)
G	위수가 q인 순환 대수군(cyclic algebraic group)
p,q	큰 소수
g	지수 기저(exponentiation base)로 G의 생성자
ri	Zq상에서 mi의해 생성되는 랜덤 값
$f(xi)$	두 노드간의 Diffie-Hellman 키 교환에 의해 생성된 값
$F(Xi)$	i번째 내부노드가 자식 노드와의 함수 $f(xi)$을 곱한 값
KEY	그룹키
IDi	i-th 멤버에 관련된 개인 정보

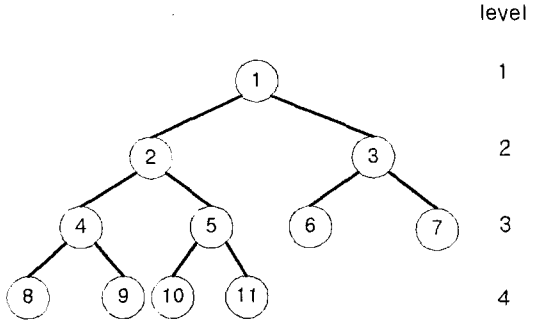
다. 이것이 잘 알려진 두 멤버간의 Diffie-Hellman 키 교환 프로토콜이다. 이 과정을 마친 두 멤버 사이에는 $f(x)$라는 비밀키(Secret Key)가 하나 생기게 된다. ID_i에 대한 정보는 그 멤버의 개인정보(이름, 주소, 전화번호 등)와 트리 상의 위치 정보에 해당되며, 이 번호(i)는 중복될 수가 없으며, 이 트리상의 유일한 번호(i)를 이용하여 트리상에서 각 그룹멤버 ID_i는 자신의 부모 노드와 자식 노드가 누구인지 쉽게 알 수 있다.

본 논문에서는 모든 그룹멤버들은 인증절차를 거쳐서 질서정연하게 구성된다고 하자. 즉, 내부 공격자는 없으며, 서로가 신뢰하는 행동만 한다고 가정한다. 그러므로 그룹멤버들은 프로토콜 규칙들을 준수하며, 그룹의 한 멤버가 값을 변조시켜 전달하거나 단독으로 생성된 값을 전달하지 않고 제안한 프로토콜들을 따른다. 모든 그룹의 멤버들은 그룹에서 일어나는 변화들을 알고 있어 그룹멤버들 각자는 트리상의 전체적인 모양을 알고 있다.

2. HGDH : Hierarchical Group Diffie-Hellman

본 절에서는 완전 이진 트리와 두 멤버간의 Diffie-Hellman 키 교환 프로토콜을 이용하여, 안전한 그룹통신에 필요한 그룹키 생성에 필요한 새로운 프로토콜(HGDH)를 제안한다. 제안한 HGDH(Hierarchical Group Diffie-Hellman) 프로토콜은 기존의 키 그래프나 키 트리와는 달리, 트리상의 모든 노드들이 그룹 멤버들로 구성되어 있다.

본 논문에서 제안한 프로토콜을 수행하기 위해서 한 그룹은 [그림 1]과 같이 논리적인 완전 이진 트리를 구성한다. 논리적으로 구성되었다는 것은 특정한 네트워크의 형상을 요구하지 않고 [그림 1]에서와 같



[그림 1] 서브그룹 내의 논리적인 구성도(logical organization in a group)

이 차례로 부여된 번호(i)에 의해서 자신의 위치를 알게 되고, 각각의 멤버들이 가진 번호(i)에 의해 자신과 연관된 부모 노드와 자식 노드가 어떤 그룹멤버인지 알 수 있다. 다시 말해, 트리를 구성하는 모든 멤버는 각자의 ID_i를 가지게 되고, ID_i는 완전 이진 트리상의 멤버의 위치 정보인 i값과 [정리 1]에 따라 부모 노드와 자식 노드를 안다.

[정리 1]에 따라 [프로토콜 1]을 수행하게 되는데, 모두 3단계로 이루어져 있다. 첫 번째 단계는 두 멤버간의 비밀키($f(x_i)$)를 생성하는 단계로, 각 그룹멤버는 자신의 부모 노드와 자식 노드간에 Diffie-Hellman 키 교환을 한다. 두 번째 단계는 그룹키를 생성하는 단계로, 트리의 각 내부 노드들이 자신의 부모 노드에게 자식 노드와 Diffie-Hellman 키 교환을 한 결과값의 곱($F(X_i)$)을 전달하여 그룹키를 만든다. 마지막 단계에서는 근 노드에서부터 단말 노드까지 두 노드 사이에만 알고 있는 비밀키($f(x_i)$)를 이용하여 안전하게 그룹키를 전송하는 단계이다.

[정리 1]

n 명으로 구성된 한 그룹에서 [그림 1]과 같이 구성되어 있다면 그룹 멤버 m_i는 다음의 정보를 알 수 있다. (단, $1 \le i \le n$)

- $i \neq 1$이면 m_i는 $\lfloor \frac{i}{2} \rfloor$의 위치에 있게 되고, 만약, $i=1$이면 i 는 근 노드이므로 부모가 없다.
- $2i \le n$이면 왼쪽 자식은 $2i$의 위치에 있게 되고, 만약 $2i > n$이면 i 는 왼쪽 자식을 가질 수 없다.
- $2i+1 \le n$이면 오른쪽 자식은 $2i+1$의 위치에 있게 되고, 만약 $2i+1 > n$이면 i 는 오른쪽 자식을 가질 수 없다.

[그림 1]과 같이 그룹이 구성되어 있을 때, 그룹키를 생성하는 과정을 예를 들어 설명한다. 첫 단계로, 각 노드는 함수 $f(x_i)$를 연산하기 위해 관련된 두 그룹멤버간에는 Diffie-Hellman 키 교환을 하게 된다. 그리하여 각각의 그룹멤버들은 부모 노드와 자식 노드간의 비밀값($f(x_i)$)을 얻게 된다. 예를 들면, m_4는 다음의 함수값들을 연산하여 부모 노드(m_2)와 자식 노드(m_8, m_9)사이에 비밀값을 공유하게 된다.

$$f(x_4)_L = g^{r_4 r_8} \bmod p ,$$

$$f(x_4)_R = g^{r_4 r_9} \bmod p ,$$

$f(x_4) = g^{r_4 r_2} \bmod p$

두 번째 단계에서는 가장 낮은 레벨의 내부 노드부터 자신의 자식 노드와 공유하고 있는 비밀값을 곱해서 자신의 부모 노드에게 안전하게 전달하고, 이것은 근 노드까지 단계적으로 이루어지며, 근 노드까지 이르게 되면 그룹키를 생성하게 된다. 예를 들면, m_4는 함수값 $F(X_4) = f(x_4)_L f(x_4)_R F(X_8) F(X_9)$을 두 그룹멤버 m_2와 m_4사이의 비밀값($f(x_4)(= f(x_2)_L)$)을 이용하여 m_2에게 안전하게 값을 전달한다. 마지막 단계에서는 두 번째 단계에서 생성된 그룹키 $F(X_1)^{(1)}$를 근 노드에서부터 단말 노드까지 부모와 자식만 알고 있는 각각의 비밀키로 안전하게 전달하게 되어, [그림 1]과 같은 n=12인 그룹은 12명만이 아는 그룹키(KEY$^{(1)}$)를 갖게 된다.

Ⅳ. 그룹멤버들에 대한 관리

그룹 내에는 그룹의 구성원이 증가하기도 하고 감소하기도 한다. 뿐만 아니라, 서로 다른 그룹들이 합한다. 이런 여러 가지 이유로 인해 그룹내에는 변화가 일어나게 되고 이런 변화들은 완전 이진 트리의 균형을 유지하는데 있어 어려움을 야기 시킨다. 또한, 그룹에 변화가 발생하면 제안한 HGDH 프로콜들은 그룹키의 안전성을 보장하기 위해 그룹멤버들은 적절한 연산을 통해 모두 협력하여 그룹키를 갱신해야 한다.

본 장에서는 그룹에 여러 변화들이 발생할 때, 완전 이진 트리의 균형을 유지하면서 그룹키를 갱신하는 방법과 그룹 멤버들의 변화에 따른 프로토콜 수행 과정을 살펴본다.

4.1 프로토콜 유지 및 그룹키 관리

어떤 그룹에 대해, 그룹멤버들이 가입(Join), 탈퇴(Leave), 병합(Merge), 분할(Partition) 등과 같은 변화가 발생하면, HGDH 프로토콜은 완전 이진 트리의 모양을 어느정도 유지하면서 그룹키를 갱신해야 한다. 그러므로 본 프로토콜은 다음과 같은 특징을 가지고 있고, 그룹에 변화가 발생하면 모든 그룹멤버들은 [RULE 1]과 [RULE 2]를 준수한다고 가정하자.

- 그룹키는 현재 그룹에 속해 있는 모든 멤버들에 의해서 생성된다. 그리고 각각의 그룹멤버는 그룹키를 생성하는데 있어 자신의 비밀값을 한번 이상 사용하고, 자신의 비밀값(r_i)는 각각의 그룹멤버가 안전하게 유지한다.
- 가입과 병합이 발생할 때, 새로운 멤버의 비밀값이 기존의 그룹키에 추가되면서 이전의 그룹키를 새롭게 갱신한다.
- 탈퇴와 분할이 발생할 때, 그룹에서 떠난 멤버들의 비밀값들은 새롭게 생성될 그룹키로부터 제거되고, 그룹에 남아있는 멤버들 중에서 적어도 한명 이상 자신의 비밀값을 바꾸어 이전 그룹키를 새롭게 갱신한다.

[프로토콜 1] Hierarchical Group Diffie-Hellman Protocol(HGDH)

단계 1 : 두 멤버간의 Diffie-Hellman 키 교환

- 모든 그룹멤버(m_i)들은 다음과 같은 함수 값들이 존재한다면 각각의 함수 $f(xi)$을 연산한다.

$f(x_i)_L = g^{r_i r_{2i}} \bmod p$, $f(x_i)_R = g^{r_i r_{2i+1}} \bmod p$, $f(x_i) = g^{r_i r_j} \bmod p$

단계 2 : 그룹키 생성

for $l = \lfloor \log_2 n \rfloor$ down to 1 *

for $i = 2^l - 1$ to 2^{l-1}

$F(X_i) = f(x_i)_L f(x_i)_R F(X_{2i}) F(X_{2i+1})$

- 널 노드(Null Node)에 해당하는 함수 값은 1로 한다.
- * : 각각의 메시지는 $f(xi)$를 이용한 암호문이다.

단계 3 : 그룹키 전달

- 근 노드에서 단말 노드까지 각 노드들 사이의 비밀값 $f(xi)$를 이용하여 $KEY(= F(X_1))$를 전송한다.

(1) $F(X_1) = g^{r_1 r_2 + r_1 r_3 + r_2 r_4 + r_2 r_5 + r_3 r_6 + r_3 r_7 + r_4 r_8 + r_4 r_9 + r_5 r_{10} + r_5 r_{11} + r_6 r_{12}} \bmod p$

[RULE 1] 트리 관리(Tree Management)
- 내부 노드가 탈퇴를 하게 되면 단말 노드들 중에서 가장 오른쪽에 위치한 노드가 그 자리를 채운다.
- 단말 노드가 탈퇴를 하게 되면, 자리를 채우지 않고 탈퇴 프로토콜을 수행한다.
- 한 멤버가 가입할 때에는 단말 노드들 중에서 가장 왼쪽부터 자리를 채운다.

[RULE 2] 키 갱신(Key Update)
- 그룹에 변화가 일어나면, 위치가 변한 노드와 변화가 일어난 노드의 부모 노드들은 반드시 자신의 비밀값을 갱신해야 한다.

[RULE 1]의 두번째 항목은 사실 완전 이진 트리에 위배되는 것이나, 그룹 변화시 생기는 복잡도(Complexity)를 줄이기 위해서 약간의 변칙을 허용한다. 그리고 [RULE 2]는 그룹키의 안정성을 보장하기 위한 기본이 된다.

한 그룹 내에 논리적으로 완전 이진 트리가 구성되어 있다면, [그림 1]와 같이 근 노드에서부터 단말 노드까지 차례대로 각각의 그룹멤버들에게 노드 번호(i)가 부여된다. 기존의 연구들과 같이 그룹을 구성하는 그룹멤버들은 트리의 구성을 알고 있어 앞서 논의한 규칙들에 따라 움직인다. 그리고 트리상의 노드 번호(i)가 [그림 1]와 같이 차례로 부여되기 때문에 단시간 내에 각 그룹멤버들은 자신의 위치를 정확히 알게 되고, 전체 트리의 일부 형태도 쉽게 짐작한다. 그룹에 변화가 발생하면 그룹멤버들은 [RULE 1]에 따라 트리 상의 논리적인 위치가 바뀌면서 위치가 바뀐 그룹멤버는 새 번호(i)를 가지게 된다. 논리적인 위치가 바뀐다는 것은 자신의 부모 노드와 자식 노드가 바뀐다는 것과 같은 의미이다. 아주 극단적인 예로, 근 노드에 위치한 그룹멤버가 어떤 이유로 그룹를 탈퇴한다면, 남아있는 그룹멤버는 [RULE 1]와 [RULE 2]의 규칙를 준수하여 트리의 균형도 유지하고 그룹키도 갱신하게 된다. 즉, 단말 노드들 중에 가장 오른쪽에 위치한 노드가 근 노드의 자리를 채우게 되고 자리가 바뀐 노드는 번호 1번을 가지게 되고, [RULE 2]에 따라 영향을 받은 노드들이 자신의 비밀값을 갱신하여 이전 그룹키를 새롭게 갱신하게 된다. 변화에 따른 자세한 프로토콜 과정은 다음 절에서 서술하고 있다.

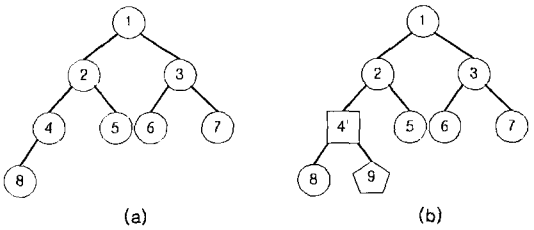
4.2 가입(Join)

가입 프로토콜은 어떤 외부자가 어떤 특정 그룹의 통신에 참여하길 원할 때 수행되어지고, 수행되는 과정은 [프로토콜 2]과 같다. 새로운 멤버들의 가입은 [Rule 1]에 따라 트리의 단말 노드에서만 일어나게 된다. 왜냐하면, [Rule 1]에 따라 내부노드의 자리가 항상 채워져 있을 뿐만 아니라, 완전 이진 트리의 특성에 따라 단말 노드들 중에서 가장 왼쪽의 빈 자리부터 채우게 된다.

[그림 2]은 가입하는 과정을 그림으로 보여준다. [그림 2]에서 원은 기존의 그룹 멤버들이고, 사각형은 기존 그룹 멤버가 자신의 비밀값을 바꾸었다는 것을 나타낸다. 그리고 오각형은 새롭게 가입한 멤버를 나타내고, 숫자는 트리상의 그룹 멤버들의 위치를 나타내는 노드 번호이다. 이것은 본 논문 전체에 적용된다.

[그림 2]에서와 같이 그룹에 새로운 멤버 m_9가 들어오게 되면, 기존 그룹 멤버들은 그룹에 변화가 일어난 것을 알게 되고, [프로토콜 2]을 수행하여 그룹키를 갱신한다. 그리하여 새로운 멤버(m_9)는 부모 노드(m_4)와 Diffie-Hellman 키 교환을 하게 된다. 이 때 m_4는 자신의 비밀값 r_4를 r'_4로 바꾸어 새로운 그룹키를 연산한다. 그런 다음, m_4는 이전 그룹키를 이용하여 새로 가입한 멤버를 제외한 기존의 그룹 멤버에게 새롭게 갱신된 그룹키를 전달한다. 그리고 새로운 멤버 m_9에게는 m_4과 m_9만 알고 있는 비밀값($f(x_4) = g^{r'_4 r_9} \bmod p$)으로 안전하게 전달하게 된다. 결과적으로 m_1에서부터 m_9까지는 다음과 같은 새로운 그룹키(KEY')를 가지게 된다.

$$KEY' = KEY(g^{r_2 r_4 + r_4 r_8})^{-1}(g^{r_2 r'_4 + r'_4 r_8})g^{r'_4 r_9}$$
$$= (KEY/g^{r_2 r_4 + r_4 r_8})g^{r_2 r'_4 + r'_4 r_8 + r'_4 r_9}$$
$$= g^{r_1 r_2 + r_1 r_3 + r_2 r'_4 + r_2 r_5 + r_3 r_6 + r_3 r_7 + r'_4 r_8 + r'_4 r_9}$$



〔그림 2〕 가입 프로토콜의 전과 후

Diffie-Hellman 키 교환 횟수를 DH라고 하고, 새롭게 Diffie-Hellman 키 교환을 한 노드를 제외하고 그룹 멤버들 중에서 자신의 비밀값을 바꾼 노드 주변에 있는 노드의 개수를 N이라고 하면 [프로토콜 2]은 아래와 같은 식이 항상 성립한다.

총 모듈라 멱승 계산 횟수 = 4 · DH + N

총 생성되는 메시지 수 = 2 · DH +N + 1

4.3 탈퇴(Leave)

어떤 그룹의 한 그룹멤버가 더 이상 그 그룹의 통신에 참여하지 않기를 원하거나 통신상의 장애로 더 이상 그 그룹 통신을 유지할 수 없을 때에는 그룹멤버의 탈퇴가 발생하게 되고, [프로토콜 3]에 따라 이전의 그룹키를 새롭게 갱신한다. 그룹 멤버가 탈퇴되는 이유는 자발적이거나 타의적인 것 등 여러 가지가 있을 수 있다.

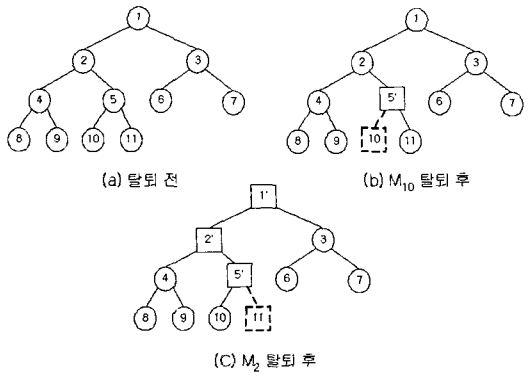
탈퇴한 멤버가 논리적으로 구성하고 있는 완전 이진 트리상의 위치에 따라 [프로토콜 3]는 두 가지 경우로 나누어진다. 예를 들면, [그림 3]의 (b)는 그룹 (a)에서 m_{10}이 탈퇴한 경우이고, (c)는 그룹 (a)에서 내부 노드인 m_2이 탈퇴한 경우이다. 먼저, 단말 노드가 탈퇴된 경우를 살펴보면, 복잡도를 줄이기 위해서 [RULE 1]에 따라 위치는 바뀌지 않고, 탈퇴한 노드의 부모 노드인 m_5는 자신의 비밀값을 r'_5로 바꾸고, 그것을 m_2와 m_{11}에게 $g^{r'_5} \bmod p$를 연산하여 전달한다. 그러면 m_2와 m_5 그리고 m_5와 m_{11}사이에는 새로운 비밀키를 생성하게 된다. 그리고 탈퇴한 멤버로부터 그룹 통신의 비밀을 보장하기 위해 그룹키를 갱신하게 된다. 이것은 m_5에서 m_2그리고 m_1까지 $F(x)$ 값을 연산하여 그룹키를 갱신한다. 그런 다음, m_1에서부터 단말 노드까지 비밀스럽게 갱신된 그룹키를 전달하게 되고, 남아있는 그룹 멤버들은 갱신된 그룹키를 가진다.[2] 만약 [그림 3]의 (b)에서와 같은 상태에서 한 멤버가 이 그룹에 가입하게 되면, 새로운 멤버는 [RULE 1]에 따라 노드 번호 10번 자리에 들어가게 된다.

[프로토콜 2] 가입 프로토콜(Join Protocol)
새로 가입할 멤버를 m_a이 하자.

- m_a은 어떤 그룹에 가입을 요구하고, [RULE 1]에 따라 단말 노드들 중에서 가장 왼쪽의 빈 자리의 번호(i)를 가지게 된다.
- m_a와 그의 부모 노드는 Diffie-Hellman 키교환을 한다. 이 때, 기존의 멤버인 m_a의 부모 노드(m_i)는 자신의 비밀값을 바꾸게 되고, $g^{r'_i} \bmod p$ 를 연산하여 자신과 관련된 노드에게 알려준다.
- m_a의 부모 노드는 새로운 그룹키를 생성하여 이전 그룹키로 이용하여 현재 그룹에게 전달하게 되고, m_a에게는 둘만이 아는 비밀키를 이용하여 안전하게 새로운 그룹키를 보낸다.

[프로토콜 3] 탈퇴 프로토콜(Leave Protocol)
탈퇴할 멤버를 m_i 라고 하자.

- m_i가 단말 노드에 위치한 경우
  - $m_{\lfloor i/2 \rfloor}$는 자신의 비밀값을 $r'_{\lfloor i/2 \rfloor}$로 갱신한다.
  - $m_{\lfloor i/2 \rfloor}$는 $g^{r'_{\lfloor i/2 \rfloor}} \bmod p$을 연산하여 그룹에 남아있는 자신의 부모 노드와 자식 노드에게 전달한다.
  - $m_{\lfloor i/2 \rfloor}$에서부터 [프로토콜 1]의 두 번째 단계이후와 비슷하게 진행된다.
- m_i가 내부 노드에 위치한 경우
  -[RULE 1]에 따라 탈퇴할 m_i의 자리를 그룹에 남아있는 단말 노드들 중에서 가장 오른쪽에 위치한 멤버가 그 자리를 채운다(m'_i).
  - m'_i와 $m'_{\lfloor i/2 \rfloor}$의 두 멤버는 자신의 비밀값을 바꾸어 새롭게 Diffie-Hellman 키 교환을 하게 된다. (이 때, 자리를 옮긴 단말 노드의 부모도 자신의 비밀값을 갱신한다. 또한, 자신의 비밀값을 갱신한 노드는 관련된 주변 노드에게 바뀐 비밀값에 멱승한 값을 전달한다.)
  -[프로토콜 1]의 두 번째 단계이후와 비슷하게 진행된다.

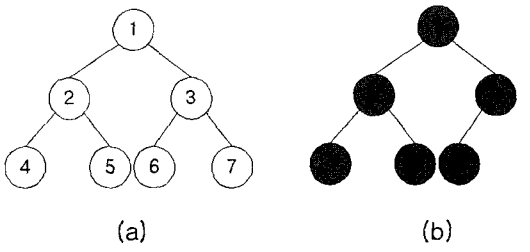


[그림 3] 탈퇴 프로토콜

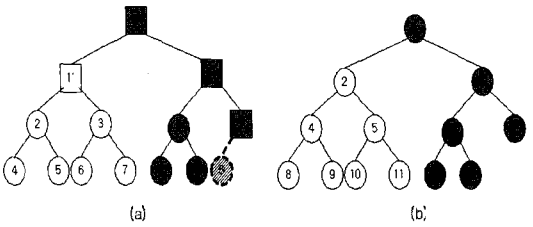
[그림 3]의 (a)에서 내부 노드인 m_2가 탈퇴할 경우는 단말 노드중에서 가장 오른쪽에 위치한 m_{11}을 m_2의 자리로 바꾸고, m_{11}는 이제 탈퇴한 m_2의 역할(m'_2)을 하게 되고, 노드 번호(i)는 11이 아니라 2가 된다. 그런 다음 m'_2와 그의 부모 노드 m_1는 자신의 랜덤한 비밀값을 바꾸어 Diffie-Hellman 키 교환을 하게 되고, m'_2와 그의 자식 노드 m_5도 마찬가지이다. 그 이후는 단말 노드가 탈퇴한 경우와 같고 새롭게 갱신된 그룹키는 다음과 같다.[3]

4.4 병합(Merge)

서로 다른 그룹들이 하나의 그룹으로 구성하려고 하거나, 네트워크가 갑자기 단절되어 한 그룹이 두 그룹으로 나누어졌다가 다시 한 그룹으로 합칠 경우 그룹에 병합이 발생한다. [그림 4]와 같이 레벨(Level)이 같은 (a)와 (b) 두 그룹이 병합할 때는 복잡도를 줄이기 위해 [그림 5]처럼 되는 것이 바람직하다. 즉, [그림 4]의 (b)에서 m_6이 자신의 비밀값을 바꾸고, [그림 5]의 (a)처럼 두 그룹의 근 노드가 되는 것이다. 그리고 [그림 4]의 (a)와 (b) 그룹의 두 근 노드는 자신의 비밀값을 바꾸어 [그림 5]의 (a)의 근 노드(m'_6)와 Diffie-Hellman 키 교환을 하고, m_6의 부모 노드(m_3)도 자신의 비밀값을 r'_3으로 바꾼다. 그런 다음, [그림 4]의 (a) 그룹의 근 노드와 (b) 그룹의 근 노드는 각각 자신의 랜덤한 비밀값을 갱신하여 이전의 (a)와 (b)의 그룹키 값을 새롭게 갱신한 그룹키를 m'_6에게 안전하게 전달한다. 이를 받은 m'_6은 [그



[그림 4] 병합 전



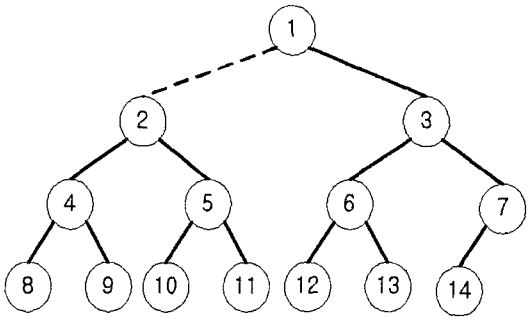
[그림 5] 병합 후

림 4]의 (a)와 (b)가 합쳐진 하나의 그룹, 즉 [그림 5]에서 사용하게 될 새 그룹키(KEY')를 연산한다. 갱신된 그룹키(KEY')를 m'_6가 지금의 자식 노드에게 안전하게 전달하면 각각은 이전의 [그림 4]의 (a)그룹키와 (b)그룹키를 이용하여 갱신된 그룹키를 한번만에 전달한다. 그리고 마지막으로 [그림 5]의 (b)와 같이 노드 번호가 새롭게 부여된다. 이와 같은 프로토콜은 합쳐질 두 그룹의 레벨이 2이상 차이 나지 않을 때 효과적이다. 레벨이 2이상이 차이가 나면 하나 하나 가입시키는 것이 적은 복잡도를 가진다. 이것은 완전 이진 트리를 만드는 것이 많은 오버헤드를 가져올 수 있기 때문에 상황에 따라 적은 복잡도를 가지게 수행되는 것이 바람직하며, 이를 위해서 각 그룹의 구성원들은 합의에 의해 복잡도를 줄이는 방향으로 진행시킬 수 있다.

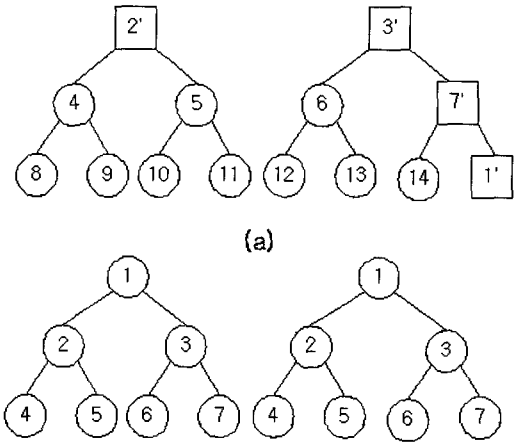
4.5 Partition

한 그룹이 둘 이상의 그룹으로 분리될 때 분할이 일어난다. 이것은 의도적으로 그룹을 나누거나 네트워크 상황이 좋지 않아서 그룹이 나누어질 수도 있다. 분할 프로토콜도 병합 프로토콜과 마찬가지로 트리상에서 분할이 일어나는 위치에 따라 복잡도를 줄

(2) $KEY = g^{r_1r_2 + r_1r_3 + r_2r_3 + r_2r'_3 + r_3r_6 + r_3r_7 + r_6r_7 + r_6r_{10} + r'_3r_{11}} \bmod p$
(3) $KEY = g^{r'_1r'_2 + r'_1r_3 + r'_2r_4 + r'_2r'_5 + r_5r_6 + r_5r_7 + r_6r_8 + r_6r_9 + r'_5r_{10}} \bmod p$



〔그림 6〕 분할 전



〔그림 7〕 분할 후

이는 방향으로 수행하는 것이 바람직하다. 만약 분할이 단말 노드 주변에서 일어난다면 이것을 탈퇴되는 경우와 동등하다. 그리고, [그림 6]처럼 근 노드 주변에서 일어나면, [그림 7]처럼 되는 것이 복잡도를 적게 한다. 자세히 설명하면, [그림 6]은 근 노드 주변에서 분할이 일어나는 경우로 [그림 7]의 (a)와 같이 두 그룹으로 나눈다. [그림 6]의 근 노드인 m_1이 [그림 7]의 (a)와 같이 자신의 랜덤한 비밀값을 바꾸고 오른쪽 그룹의 단말 노드로 들어간다. 그리고 m_2, m_3, m_7는 각각 자신의 비밀값을 바꾸고, m'_7는 새로 들어온 단말 노드와 Diffie-Hellman 키 교환을 하게 된다. 마지막으로 두 그룹의 근 노드에 위치한 노드 m'_2와 m'_3에서 단말 노드까지 단계적으로 갱신된 그룹키를 안전하게 전달하게 되고, [그림 7]의 (b)와 같이 노드 번호가 각각 다시 부여되어 서로 다른 두 개의 그룹으로 나누어지게 된다.

V. 암호적 속성(cryptographic properties)

[정리 2]

p가 큰 소수이고 g를 $GF(p)$의 원시 근이라 할 때, $y \neq 0 \bmod p$인 임의의 y에 대해 $y = g^x \bmod p$를 만족하는 $x \in Z_{p-1}$이 존재한다.

[가정 1]

소수 p, Z^*p상의 생성자 g, $g^a \bmod p$, $g^b \bmod p$가 주어졌을 때, $g^{ab} \bmod p$를 찾는 문제를 Diffie-Hellman problem라고 한다.

[정리 2]

에서 p, g, y가 주어졌을 때, x를 구하는 문제는 매우 어려우며, 이를 이산대수 문제라고 한다. Diffie-Hellman 키 교환은 [가정 1]에서와 같은 DHP(Diffie-Hellman problem)를 기반하고 있고, DHP는 이산대수 문제의 어려움으로 귀착된다. 그러나, DHP의 어려움 정도가 이산대수 문제의 어려움과 동등한지를 증명하는 것은 아직 알려지지 않았고 하나의 큰 연구대상으로 남아있다. 본 논문에서 제안한 프로토콜은 polynomial 시간내에서 이산대수 문제를 푸는 어려움에 기인한다. 즉, 이산대수 문제를 푼다는 것은 DHP도 풀 수 있다는 것이다.[8] 본 논문에서 제안한 프로토콜은 그룹멤버 각자가 실행시점에서 비밀스럽게 랜덤한 비밀값을 선택하기 때문에 이 비밀값은 그룹멤버 자신만 알 수 있고 공개되어 있는 값만으로는 그 비밀값을 알 수 없다. 따라서, Diffie-Hellman 문제가 어렵다면, 본 논문에서 제안한 HGDH(Hierarchical Group Diffie-Hellman) 프로토콜에 의해서 생성된 키는 비밀성(Privacy)을 보장하며, 외부에 알려진 일부 정보를 이용하여 그룹키를 추측하는 수동적인 공격(Passive Attack)에도 안전한다. 그리고, 앞서 모든 그룹멤버들이 신뢰할 수 있는 행동만 한다고 가정하였기 때문에, 내부자에 의한 공모나 공격은 없다. 마지막으로 본 논문에서 제안한 HGDH 프로토콜은 그룹키의 안전성을 위해 다음과 같은 암호적 속성(Cryptographic Properties)들을 만족한다.[12]

- 그룹키 비밀성(Group Key Secrecy) : 가장 기본적인 속성으로, 어떤 특정 그룹에서 사용되었던 혹은 사용 중인 그룹키는 그룹의 구성원들만이 알 수 있어야 한다.

- 전방 비밀성(Forward Secrecy) : 그룹 멤버의 변화로 어떤 멤버가 더 이상 그룹 멤버가 아니거나 수동적인 적(passive adversary)이 옛 그룹키의 서브셋을 알더라도 앞으로 그룹에서 사용할 새로운 그룹키에 대해서 알 수 없어야 한다.
- 후방 비밀성(Backward Secrecy) : 그룹 멤버의 변화로 새로운 멤버가 그룹에 가입되거나 수동적인 적이 그룹키의 서브셋을 알더라도 이전에 그룹 통신에 사용된 그룹키에 대해서 알 수 없어야 한다.
- 키 비밀성(Key Secrecy) : 가장 중요한 속성으로 그룹키를 이루는 각각의 인자는 그룹 멤버들 각자가 비밀스럽게 생성한 랜덤값이어야 하며, 외부에 노출되지 않아야 한다.

본 논문에서 제안한 프로토콜의 그룹키 비밀성은 앞서 설명한 것과 같이 이산대수 문제의 어려움에 기반하며, 모든 그룹 멤버가 스스로 안전하게 비밀값을 랜덤하게 선택함으로 키 비밀성을 보장한다. 게다가, 그룹에 변화가 발생할 때마다 그룹 멤버들 중에서 적어도 한명이상의 멤버가 자신의 비밀값을 갱신하기 때문에 항상 전방 비밀성과 후방 비밀성이 보장하며, 그룹키 비밀성은 전방 비밀성과 후방 비밀성에 포함된다.

Ⅵ. 복잡도 비교 및 결론

본 절에서는 본 논문에서 제안한 HGDH 프로토콜을 이전의 여러 가지 관련 연구들에 대해 통신량(Communication)과 연산량(Computation) 측면에서 비교하고 분석한다. GDH.1,[10] GDH.3,[10] Hypercube,[2] Octopus,[2] 그리고 트리에 기반한 BD[1]는 브로드캐스트를 사용하지 않은 프로토콜이며, GDH.2,[10] TGDH,[12] STR,[13] BD[1]는 브로드캐스트를 이용한 프로토콜이다. 이들 각 프로토콜들은 두 멤버간의 Diffie-Hellman 키 교환을 서로 다른 방법을 이용하여 그룹으로 확장시킨다. 본 논문에서 제안한 HGDH 프로토콜도 두 멤버간의 Diffie-Hellman 키 교환을 다른 방법을 이용하여 그룹키를 생성하며, GDH.1과 GDH.3, Hypercube 그리고 Octopus 프로토콜과 비교 대상이 되며, 근 노드에서 단독으로 키를 만들어 분배하는 BD[1]는 비교 대상에서 제외시킨다. [표 2]과 [표 3]은 이들 프로토콜의 복잡도를 비교한 표이다.

제안한 HGDH 프로토콜은 그룹키를 만드는데 있어 각각의 그룹 멤버가 비밀스럽게 선택한 비밀값의 사용정도가 똑같지 않다. 근 노드가 2번, 내부 노드들이 각각 3번 그리고 단말 노드들이 각각 1번씩 자신의 랜덤한 비밀값을 그룹키를 생성시키는데 기여하게 된다. [표 2]는 처음으로 그룹에서 그룹키를 설정하는데 드는 복잡도이다. 다른 방법들에 비교해 볼 때, 본 논문에서 제안한 프로토콜은 브로드캐스트를 사용하지 않고, 그룹 멤버들 사이에 단계적으로 메시지를 전달함에도 불구하고 통신량의 측면에서 나쁜 복잡도를 나타내지 않는다. 연산량 측면에서 보면, 브로드캐스트를 사용하지 않은 프로토콜중에서는 본

〔표 2〕 초기 그룹키 생성에 관한 복잡도 비교

	Communication					Computation			
	Round	Messages	T.Bandwidth	Sent*	Received*	T.Exp	Exp*	T.Mul	Mul*
HGDH	$1+2\log_2 n$	$2n-L$	$4(n-1)-L$	$(1,5,3)$	$(2,4,2)$	$4n-M-1$	$(2,4,3)$	$n-L$	$(0,1,1)$
2^dHypercube	d	nd	nd	d	d	$nd+2^d$	$d+1$	-	-
2^dOctopus	$2+d$	$3(n-2^d)+2^d d$	$3(n-2^d)+2^{d-1}d$	$d+O$	$d+O$	$2(n-2^d)+2^{d-2}d$	$d+(n-2^d)$	-	-
GDH.1	$2(n-1)$	$2(n-1)$	$n(n-1)$	2	2	$\frac{n^2+3n}{2}-1$	$i+1$	-	-
GDH.3	$n+1$	$2n-1$	$3(n-1)$	2	3	$5n-6$	$4(2=Mn-1,$ $n-1=Mn)$	-	-
BD	2	$2n$	$2n$	2	$n+1$	$3n$	3	n^2	n
GDH.2	n	n	$\frac{n^2+3n}{2}-3$	1	2	$\frac{n^2+3n}{2}-1$	$i+1$	-	-
TGDH	$1+\log_2 n$	$2(n-1)$	$2n-1$	1	n	$n(\log_2 n+1)$	$\log_2 n+1$	-	-
STR	n	$2n$	$2n$	$1(M_1=n)$	n	n^2	n	-	-

(단, * : per member , M=2l , L=2l-1, l= level of tree , O=2d을 중심으로 있는 그룹 멤버들의 수, (x, y, z)=(leaf node, internal node, root node), T.=total, Exp=Exponentiations, Mul=Multiplications 임.)

논문에서 제안한 HGDH 프로토콜은 좋은 복잡도를 보이고 있다. 브로드캐스트를 사용하는 TGDH[12]와 STR[13] 프로토콜은 좋은 복잡도를 가지고 있지만, 트리의 높이에 따라 실제 비용이 달라진다. 그러나, 제안한 HGDH 프로토콜은 그룹 멤버들 각자가 일정한 지수승과 곱셈 연산 비용을 가지며, 그룹의 크기가 변하더라도 그룹멤버 각자가 가지는 연산량은 일정하여, 연산량 측면에서 가장 적은 비용으로 그룹키를 설정할 수 있는 프로토콜이다. 브로드캐스트를 사용한 BD 프로토콜에서도 일정한 연산 비용을 가지고 있지만, 그룹키를 만들기 위해 곱셈 연산에 그룹 멤버들 각자는 O(n)의 비용이 소요되며, 그룹의 크기에 따라 연산량은 선형적으로 변한다.

모듈러 멱승 연산량을 줄이는 문제는 한 연구 분야로 자리잡고 있을만큼 중요하다. 그러나, [표 4.1]에서 보면, 키 트리를 사용한 TGDH[12]과 STR[13]는 효율적인 프로토콜이긴 하나, 나머지 프로토콜에 비해 각 그룹 멤버에게 많은 모듈러 멱승 연산량을 요구한다. 본 논문의 HGDH 프로토콜은 그룹 멤버들 각자가 수행할 모듈러 멱승 연산과 곱셈 연산의 양은 상수번으로 일정하다. 즉, 단말 노드일 경우는 모듈러 멱승 연산은 2번하고, 내부노드일 경우는 4번 그리고 근노드일 경우는 3번을 한다. 또한 곱셈연산은 각 내부노드에서만 1번의 비용이 든다.

[표 3]는 그룹에 발생되는 변화중 가입과 탈퇴에 따라 소요되는 복잡도이다. 표에서는 가입과 탈퇴에 대해서만 언급하였지만, 병합와 분할은 가입과 탈퇴의 반복으로 볼 수 있다. 그렇지만, HGDH 프로토콜은 4절에서 보인 병합과 분할의 경우는 오히려 다른 프로토콜보다 좋은 복잡도를 가진다. 그리고, 특정 네트워크의 형태를 요구하는 Hypercube와 Octopus 프로토콜은 그룹 변화에 부적절한 프로토콜이기 때문에 비교 대상에서 제외시킨다.

[표 3]에서 HGDH 프로토콜에서는 그룹 멤버의 탈퇴가 다소 많은 라운드와 메시지가 발생하지만, 브로드캐스트 한번이 유니캐스트(Unicast) n번과 같다는 것을 감안할 때, 메시지 측면에서 다른 프로토콜에 비해 뒤지지 않는다. 그리고 그룹 멤버들 각자는 그룹 크기에 상관없이 항상 일정하게 적은 연산 비용이 가진다. TGDH[12] 프로토콜과 STR[13] 프로토콜은 변화가 일어나는 위치에 따라 아주 많은 영향을 받는다. 특히, STR 프로토콜은 점점 낮은 레벨에서 변화가 발생할수록 복잡도가 커진다. [표 3]는 평균값을 나타내었다. 본 논문의 HGDH 프로토콜도 변화가 발생하는 트리상의 위치에 따라 발생하는 메시지 수와 전체 연산 비용은 다소 달라지만, 각각의 멤버가 가지는 연산 비용은 일정하다. 그리고 HGDH 프로토콜에서는 내부 노드보다 단말 노드에서 탈퇴가 발생할 때 연산량 측면에서 최소값을 가진다.

최근에 안전한 그룹 통신은 많은 연구 대상이 되어 오고 있다. 안전한 그룹 통신에서 중요한 문제는 안전하고 효율적으로 그룹키를 설정하고 관리하는 것이다. 신뢰할 수 있는 서버가 있는 방법이 연산측면에서 적절할 수 있으나, 서버를 이용하는 방법은

〔표 3〕 그룹 멤버의 변화에 따른 복잡도 비교

		Communication				Computation	
		Round	Messages	Unicast	Broadcast	T.Exp	Exp*
HGDH	Join	*2*	*5*	*1*	*1*	*5*	*2*
	Leave	*1+2h*	*n+h+5*	*n+h-1*	*0*	*[4, 11]*	*[1, 2]*
GDH.3	Join	*4*	*n+3*	*n+1*	*2*	*n+3*	*4(n-1 for M_n)*
	Leave	*1*	*1*	*0*	*1*	*n-1*	*3(n-1 for M_n)*
BD	Join	*2*	*2(n+1)*	*0*	*2(n+1)*	*3n*	*3*
	Leave	*2*	*2(n-1)*	*0*	*2(n-1)*	*3n*	*3*
TGDH	Join	*2*	*3*	*0*	*3*	$\frac{3h}{2}n$	$\frac{3h}{2}$
	Leave	*1*	*1*	*0*	*1*	$\frac{3h}{2}n$	$\frac{3h}{2}$
STR	Join	*2*	*3*	*0*	*3*	*n+3*	*4*
	Leave	*1*	*1*	*0*	*1*	$(\frac{3h}{2}+2)n$	$\frac{3h}{2}+2$

(단, h = height of tree, [x, y]= [minimum, maximum], Exp = modular Exponentiations 임.)

단일 장애점 문제가 있다. 그래서 서버의 도움없이 모든 그룹멤버들이 함께 협동하여 그룹키를 만드는 프로토콜은 많은 관심의 대상이 되어 오고 있다. 이런 프로토콜은 그룹에 변화가 발생하면 그룹 멤버들이 협력하여 최신의 그룹키로 갱신하나, 대부분의 프로토콜은 그룹멤버 각자에게 그룹키 연산에 많은 부담을 안겨 주고 있다. 그러나 본 논문에서 제안된 HGDH(hierarchial group Diffie-Hellman) 프로토콜은 그룹을 구성하는 각각의 그룹멤버들에게 항상 일정한 연산량을 요구한다. 한 그룹 멤버의 연산량은 그룹의 크기에 독립적이며, 그룹에 변화가 발생하여 그룹키를 갱신해야 할 때도 각 그룹멤버는 항상 일정한 연산량을 가진다. 이 점은 적은 메모리나 적은 파워을 가지는 이동 통신망(ad-hoc network)을 구성하거나 적은 리소스를 가지고 그룹 통신을 하고자 하는 장치들에게 적은 연산 비용으로 안전한 그룹 통신을 할 수 있는 방법이 될 수 있다.

본 논문에서는 한 그룹을 이루고 있는 멤버들은 모두 믿을 수 있는 행동만을 한다고 가정하였으나, 향후 본 논문에서 제안한 HGDH 프로토콜에 인증(message authentication)과 키 무결성(key integrity) 등에 관한 다양한 암호학적 기능을 추가하여 더욱 향상된 프로토콜로 발전시킬 필요가 있다. 키의 무결성을 해결하는 간단한 방법으로는 메시지 전달시 해쉬값을 붙여서 보냄으로써 키의 무결성을 확인할 수 있으나, 더 정교한 고찰이 필요함으로 향후 과제로 남겨둔다.

참 고 문 헌

[1] Burmester, M. V. D. and Desmedt, Y., "A Secure and Efficient Conferen ce Key Distribution System", In A. D. Santis Ed., Advances in *Crypto-EuroCrypt'94*, pp.275~286, 1994.

[2] Klaus Becker and Uta Wille, "Communication Complexity of Gro up Key Distribution", *In Proc 5th ACM Conference on Computer and Communications Security*, San Francisco, CA USA, November, pp. 1~6, 1998.

[3] Just, M. and Vaudenay, S., "Auth enticated Multiparty Key Agree ment", In Advances in *Cryp tolo gy-Asiacrypt'96*, pp.36-49, 1996.

[4] L. R. Dondeti, S.Mukherjee and A. Samal, "A Dual Encryption Protocol for Scalable Secure Multicasting", In *the Fourth International Sym posium on Computer and Comm unications*, July 1999.

[5] Ohad Rodel, Kennneth P. Birman and Danny Dolev, "Using AVL Trees for Fault Tolerant Group Key Man agement", In Nov. 2000.

[6] G. Ateniese, M. Steiner & G.Tsudik, "Authenticated Group Key Agree ment and Friends*", In proceedings of *the 5th ACM Conference on Com puter and Communication Security*, November 2~5, 1998.

[7] S. Mittra, "Iolus: A Framework for Scalable Secure Multicasting", In Proceedings of *ACM SIGCOMM'97*, September, pp.277~288, 1997.

[8] Alfred J. Menezes, Paul C. van Oorschot and Scott A. Vanstone, "Handbook of Applied Crypto graphy", CRC Press LLC, p.113, 1997.

[9] Wong, C.K., Gouda, M. and Lam, S.S., "Secure Group communication using Gey Graphs", In *ACM SIGCOMM*, pp.68~79, 1998.

[10] Michael Steiner, Gene Tsudik and Michael Waidner, "Diffie-Hellman Key Distribution Extended to Gro up Communication", In *3rd ACM Conference on Computer and Com munications Security*, pp.31~37, New Delhi, India, March 1996.

[11] O. Rodeh, K. P. Birman and D. Dolv, "Optimized Goup Rekey for Group Communication Systems", In Proceedings of *Network and Distr ibuted System Security Sympo sium(NDSS'00)*, February 2000.

[12] Y. Kim, A. Perrig and G. Tsudik, "Simple and Fault-tolerant Key Agreement for Dynamic Colla borative Groups", In *ACM CCS 2000*, November 2000.

[13] Y. Kim, A.Perrig and G. Tsudik, "Communication-efficient Group Key Greement", In Proceedings of *IFIP SEC 2001*, June 2001.

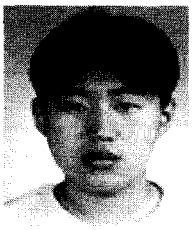
[14] G. Ateniese, M. Steiner and G. Tsu dik, "New Multiparty Authentica tion Services and Key Agreement Protocols", *IEEE Jour nal of Selected Areas in Comm unication* , vol.18, March 2000.

[15] Michael Steiner, Gene Tsudik and Michael Waidner, "CLIQUES: A New Approach to Group Key Agreement", In Proc. *18th Inter national Conference on Distributed Computing Systems*, Amsterdam, The Netherlands, May 1998. IEEE Computer Soceity Press, pp.380~387.

〈著者紹介〉

박 영 희 (Younghee Park) 정회원
2000년 2월 : 경성대학교 컴퓨터공학과 졸업
2003년 2월 : 한국과학기술원 전산학과 석사 졸업
2003년 3월 : 국가보안기술연구소
<관심분야> 정보보호, 네트워크 보안

정 병 천 (Byungchun Chung) 정회원
1998년 2월 : 성균관대 정보공학과 졸업
2001년 2월 : 한국과학기술원 전자전산학과 석사
2001년 3월 : 한국과학기술원 전자전산학과 박사과정
<관심분야> 정보보호, 네트워크 보안



이 윤 호 (YounHo Lee) 정회원
2000년 2월 : 한국과학기술원 전산학과 학사
2002년 2월 : 한국과학기술원 전자전산학과 석사
2002년 3월 : 한국과학기술원 전자전산학과 박사 과정
<관심분야> 암호학, 네트워크 보안

김 희 열 (Heeyoul Kim) 정회원
2000년 2월 : 한국과학기술원 전산학과 학사
2002년 2월 : 한국과학기술원 전자전산학과 석사
2002년 3월 : 한국과학기술원 전자전산학과 박사 과정
<관심분야> 암호학, 네트워크 보안

이 재 원 (Jaewon Lee) 정회원
1997년 2월 : 한국과학기술원 전산학과 학사
1999년 8월 : 한국과학기술원 전자전산학과 전산학전공 석사
1999년 9월 : KAIST 전자전산학과 전산학전공 박사과정
<관심분야> 타원곡선 암호, 암호 프토로콜, S/W 보호

윤 현 수 (Hyoonsoo Yoon) 정회원
1979년 : 서울대학교 전자공학과 학사
1981년 :한국과학기술원 전산학과 석사
1981년~1984년 : 삼성전자 연구원
1988년 : 오하이오 주립대학 전산학 박사
1988년~1989년 : AT&T Bell Labs. 연구원
1989년 : 한국과학기술원 전산학과 교수
<관심분야> Adhoc망, 네트워크 보안, 암호학, 상호연결 네트워크