Ad-Hoc 네트워크에서 링크 장애를 고려한 효율적인 키 협정 방법

이 영 준[†] · 민 성 기[††] · 이 성 준[†††]

요 약

Ad-Hoc네트워크에서는 기존의 인프라를 사용하지 않기 때문에 공개 키 기반 구조 또는 제삼자 키 관리 서비스를 지원하지 않는다. 따라서 여러 유형의 보안문제가 발생할 수 있다. 그래서 보안 문제를 해결하기 위한 방법인 키 협정(key agreement)에 대하여 많은 프로토콜들이 제안되어 왔다. 가장 대표적인 것이 디피 헬만(Diffie-Hellman)이 제안한 프로토콜이다. 그러나 이 방법은 두 명의 사용자간에서만 사용될 수 있다. 이 논문에서는 디피 헬만 방법을 확장하여 다자간에도 사용될 수 있는, 그룹 키 협정에 대하여 알아보고, 그룹 키 협정 진행 중에 링크 장애가 발생했을 때 그룹 키 협정을 성공적으로 수행하기 위한 효율적인 방법을 제안하였다.

Efficient Fault Tolerant Key Agreement for Ad-Hoc Networks

Young-Jun Lee[†] · Sung-Gi Min[††] · Sung-Jun Lee[†††]

ABSTRACT

Ad-Hoc network is wireless network architecture without infrastructure. We encounter new types of security problems in Ad-Hoc networks because such networks have little or no support from infrastructure. Thus, wireless communications need security mechanisms in order to guarantee the integrity and the privacy of the communication, as well as the authentication of the entities involved. Many practical systems have been proposed. The most familiar system is the Diffie-Hellman key distribution system. This algorithm allows the establishment of a cryptographic secret key between two entities. If more than two users want to compute a common key, then a group key agreement system is used. This paper discusses several group key agreement systems and presents two efficient fault tolerant methods to perform successful group key agreement.

Keywords : Ad-Hoc Network, Group Key Agreement

1. 서 론

1970년대 초 Mobile Packet Radio라 불리는 무선 네트워크가 제안된 이후로 다양한 형태의 시스템들이 개발되어 왔다. 이런 무선 네트워크는 크게 인프라 기반의 네트워크와 인프라 기반이 없는 네트워크로 분류할 수 있는데, 인프라가 없는 네트워크를 Ad-Hoc 네트워크라고 부른다. Ad-Hoc 네트워크는 무선 통신이 가능한 둘 이상의 장치들의 집합으로, 이런 장치들은 통신 영역 안에서 또는 영역 밖에 있는 다른 장치들과 통신할 수 있어야 한다. 특히 Ad-Hoc 장치는

† 종신회원: 한국교원대학교 컴퓨터교육과 교수
†† 고려대학교 컴퓨터학과 교수
††† LG 전자 DM연구소 연구원
논문접수: 2003년 12월 12일, 심사완료: 2004년 1월 8일
* 본 논문은 2003년 한국교원대학교 학술연구비에 의하여 지원되었음

영역 밖에 있는 다른 장치들과의 통신을 위해서는 패킷을 중계하거나 전송시킬 수 있는 중간 노드의 역할을 할 수 있어야 한다. 이런 Ad-Hoc 네트워크는 그 특성상 임시 구성 망이나 재해, 재난 지역이나 전쟁터와 같이 기존의 인프라 시설을 이용할 수 없는 곳에서 사용되는 것으로 인식되어 왔다.

Ad-Hoc 네트워크는 라디오 주파수를 전송매체로 하여 정보를 교환하는 무선 통신이므로, 통신 영역 안에 위치해 있는 다른 노드들도 정보를 들을 수 있는 위험성이 있다. 그래서 무선 통신은 통신의 무결성(integrity), 기밀성(privacy), 인증(authentication)을 보장해주기 위한 보안 메커니즘이 필요하게 되었다. 특히 Ad-Hoc 네트워크 내의 모든 장치는 인프라가 제공되지 않는 제한된 환경에서 통신 영역 내에 있는 장치들과 직접적인 통신을 만들 수 있어야 한다. 이렇게 인프라를 가지지 않는 환경에서의 직접적인 통신의 보안을 위하여 그룹의 참가자들은 메시지를 암호화하거나, 암호를 해독할 때 사용되어지는 키를 생성해야 한다. 키 협정은 키를 생성해서 공유하는 방법들 중 한 가지이다.

일반적으로 Ad-Hoc 네트워크에서 그룹 내의 구성원들은 구성원간 그룹의 생성, 해체 속도가 빠르며, 또한 구성원들은 고정되어질 수 있지만 그들은 움직이기도 한다. 또 라디오 주파수는 유선에 비해 데이터 손실이 많고, 구성원들이 움직이기 때문에 Ad-Hoc 네트워크의 연결은 보통 불안정하다. 이런 Ad-Hoc 장치는 네트워크의 범위를 벗어나거나, 여러 가지 장애에 의해, 구성원들 사이의 링크 연결은 끊어질 수 있다. 이 논문에서는 이렇게 링크 장애가 발생했을 때, 이 장애를 보완하여 키 협정을 수행하는 알고리즘을 제안한다.

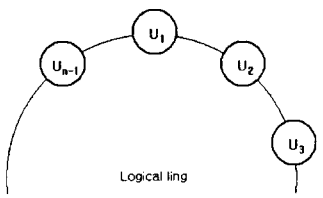
이 논문의 구성은 다음과 같다. 제2장에서는 그룹 키 협정을 위한 여러 가지 프로토콜과 그 효율성이 논의 된다. 제3장에서는 링크 장애에 대처할 수 있는 새로운 효율적인 그룹 키 협정 방법을 제안한다. 4장에서는 이 논문에서 제안한 알고리즘과 기존의 알고리즘에 대하여 그 효율성을 비교하고, 마지막으로 5장에서는 결론과 미래 연구 방향이 제시된다.

2. 그룹 키 협정 프로토콜

양자간 디피-헬만(Diffie-Hellman) 키 교환 방법이 발표된 이후로, 이를 확장하여 다자간에도 적용 가능한 '그룹 키 협정(Group Key Agreement)'에 관한 많은 프로토콜들이 제안되어왔다. 이 장에서는 다양한 프로토콜들 중에 널리 알려진 몇 개의 프로토콜들을 소개하고 그 프로토콜의 특성과 효율성을 설명한다.

2.1 Ingemarsson et al. 프로토콜

이 프로토콜은 Ingemarsson et al.이 제안한 것이다[2]. n명의 그룹 안에 있는 구성원들은 (그림 1)과 같이 논리적인 링을 형성한다.



(그림 1) Ingemarsson 프로토콜

각 라운드(round)에서 참가자들은 이전 라운드에서 받은 값을 자신의 비밀 키 값으로 제곱하고, 그것을 다음 노드로 값을 보낸다. 더 구체적으로 이야기하면 R_v는 random integer이고, $S^{(j)}(I)$는 j차수인 R_v값의 조합의 합이라고 할 때 $S^{(j)}(I)$는 다음과 같은 값을 가지게 된다.

예: $S^{(1)}(\Omega)=\sum_{i=0}^{M-1}R_i, S^{(M)}(\Omega)=\prod_{i=0}^{M-1}R_i,$

$\{\Omega \doteq (0,1,2,\cdots,M-1)\}$

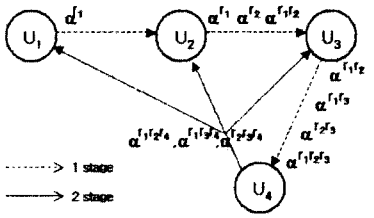
$S^{(2)}(1,3,4)=R_1R_3+R_1R_4+R_3R_4$

그래서 모든 메시지들은 $\{\alpha^{S^{(1)}()}, \alpha^{S^{(2)}()}\} \bmod p$의 형태로 전송되고 키는 $K^{(j)}=\alpha^{S^{(j)}(\Omega)} \bmod p$를 가지게 된다. 각 라운드에서 n명의 그룹일 경우에 n개의 메시지를 교환하게 된다.

2.2 GDH.2 프로토콜

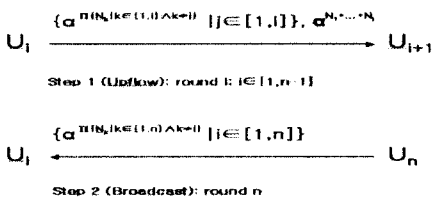
이 프로토콜은 Steiner et al[5, 6]이 제안한 것이다. Steiner, Tsudik, Waidner은 GDH.1,

GDH.2를 제안했다[6]. GDH.1의 라운드(round) 수를 줄이기 위하여 제안한 것이 GDH.2이다.



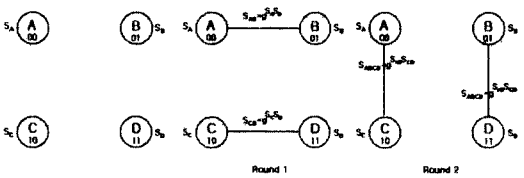
(그림 2) GDH.2 (n=4)

GDH.2 프로토콜에서는 마지막 과정에서 한번의 브로드캐스트(broadcast)메시지를 필요로 한다. 이것의 위상은 linear chain이다.(그림 2) 이 프로토콜은 마지막 단계에서 마지막 그룹 참가자인 U_n이(group controller라고 불린다.) 자신의 값을 덧붙여서 그룹 키를 만든 다음, 브로드캐스트 한다. 이 프로토콜은 메시지의 수를 줄이도록 설계되어졌으며, 다음과 같이 수행된다.



2.3 하이퍼큐브(Hypercube) 프로토콜

이 프로토콜은 Becker와 Wille[4]에 의해 제안되었고, 라운드의 수를 줄이기 위해 만들어졌다. (그림 3)과 같이 네 개의 A, B, C, D 노드로 구성되어진 네트워크가 있다. 키 협정을 하기 위하여 먼저 노드 A, B가 키를 교환하고, 이 때 동시에 노드 C, D도 키를 교환한다. 그 다음 라운드에서는 A는 C와, B는 D와 동시에 키를 교환하게 된다. 그래서 A, B, C, D는 $S_{ABCD} = g^{S_{AB}S_{CD}}$를 가지게 된다. 이 프로토콜은 만약에 노드가 2^d개라면 d 라운드 안에 실행하게 된다.



(그림 3) 하이퍼큐브 프로토콜 (n=4)

2.4 그룹 키 협정 프로토콜의 효율성 비교

Eric Ricardo Anton, Otto Carlos Muniz Bandeira Duarte[7]은 각 그룹 키 협정 프로토콜에 대하여 노드의 수에 따른 메시지의 수와 기하급수적인 연산들의 크기를 측정해보았는데, Ingemarson et al. 프로토콜은 노드의 수가 증가 할수록 메시지의 수와 기하급수적 연산의 크기가 급격하게 증가되는 것을 보여주었고, 반면 GDH.2, 하이퍼큐브 프로토콜은 서서히 증가함을 보여주었다.

하이퍼큐브 프로토콜은 가장 작은 라운드 수와 비교적 작은 메시지의 수를 가진다. 그리고 무선 네트워크에서는 메시지의 길이가 짧을수록 효율성이 좋은데 GDH.2의 메시지에는 여러 개의 공개키가 포함되어있어 메시지의 길이가 긴 반면에 하이퍼큐브 프로토콜은 메시지의 길이가 한 개의 공개키만을 가지므로 짧은 편이기에 하이퍼큐브 프로토콜이 가장 효율적인 방법이다.

이 논문에서는 하이퍼큐브 프로토콜을 근간으로 하여 링크 장애 발생시에도 사용 가능한 효율적인 방법을 제시 할 것이다.

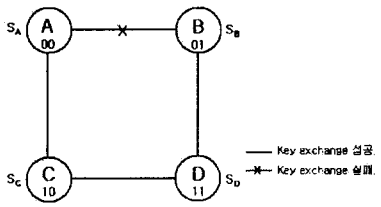
	GDH.2	Hypercube Protocol
rounds	n	d
messages	n	nd
exchanges	n	$\frac{nd}{2}$
synchronous rounds	n	d
broadcast	1	사용안함

3. 링크 장애를 고려한 효율적 하이퍼큐브 프로토콜

여기에서는 링크(link) 장애가 발생했을 경우도 수행될 수 있는 확장된 하이퍼큐브 프로토콜을 제시한다. 2절에서 논의된대로 하이퍼큐브 프로토콜이 그룹 키 협정 프로토콜 중 효율적인 방법이므로 이를 근간이 되는 프로토콜로 선택하였다. 여기에서는 Asokan, Ginzboorg[1]이 제안

한 디피-헬만 키 교환을 사용한 패스워드기반의 하이퍼큐브 프로토콜을 사용하였다.

다음은 노드 A, B, C, D가 하이퍼큐브 프로토콜을 이용하여 키 협정을 통해 키를 공유하려고 하지만, 노드 A와 노드 B사이에 링크 장애가 발생하는 경우이다.

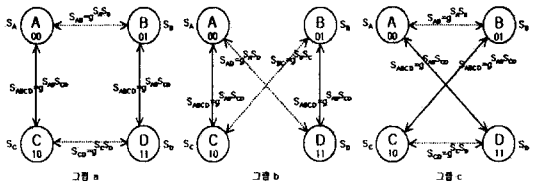


(그림 4) 하이퍼큐브 프로토콜에서의 링크 장애

(그림 4)에서와 같이 A와 B사이에 링크 장애 발생하였기 때문에 노드 A와 B는 키에 관한 메시지 교환을 할 수 없게 된다. 이 때 노드 A와 B는 노드 자체에 문제가 발생한 것이 아니므로, 다른 링크를 통해 키 협약을 계속 진행해 나갈 수가 있다. 하나의 링크 장애가 발생했을 때, 링크 장애로 인하여 키 협약을 처음부터 다시 수행하는 것은 오버헤드가 크다. 그렇다고 해서 링크가 깨진 참가자에 대해서 무조건 그룹에서 제외시키는 것도 문제가 있다. 그래서 다른 링크를 이용하기 위하여 패턴을 변형하여 키 협정을 행함으로서 이런 문제들을 해결할 수 있다.

3.1 패턴의 변형을 이용하여 링크 장애를 보완하는 알고리즘

하이퍼큐브 프로토콜에서 키 협약을 수행할 수 있는 링크의 패턴은 여러 가지가 있을 수 있다. (그림 5)은 키 협약을 수행할 수 있는 여러 가지 패턴들을 보여준다. (그림 5)에서 보는 바와 같이 노드 A, B, C, D는 여러 패턴으로 키 협약을 수행해 나갈 수 있다. (그림 5a)와 같은 경우는 A-D 또는 B-C사이에, (그림 5b)는 A-B 또는 C-D사이에, 그리고 (그림 5c)는 A-C 또는 B-D사이에 링크 장애가 발생했을 때 링크와는 관계없이 아무 문제없이 키 협약을 수행할 수 있다. (그림 5)에서 점선은 네 개의 노드가 하이퍼큐브 프로토콜을 실행할 때, 1 라운드에서 병렬로 행해지는 키 협약을 말하고, 실선은 2 라운드에서의 키 협약이 수행되어지는 것을 나타낸다. 이렇게 여러 패턴으로 진행되어지는 노드간의 키 협약이 있기 때문에, 우리는 하나의 링크 장애의 발생과 무관하게 키 협약을 할 수 있다.

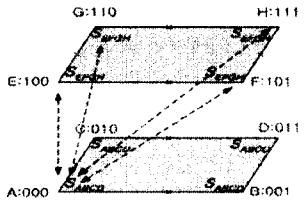


(그림 5) 여러 가지 패턴의 키 협약들

(그림 5)에서 만약 하이퍼큐브 프로토콜을 이용해서 아무런 문제없이 키 협약을 하게 된다면 (그림 5-a)와 같이 1 라운드에서 A와 B, C와 D가 병렬로 키 교환을 하게 되고, 2 라운드에서는 A와 C, B와 D가 키 교환을 하게 되어 네 개의 노드는 $S_{ABCD} = g^{S_{AB}S_{CD}}$를 공유하게 된다.

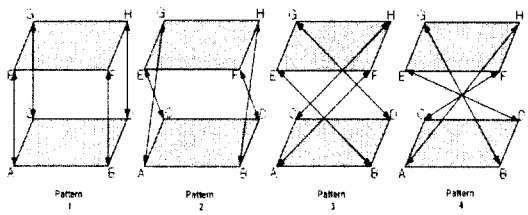
그러나 어느 노드 사이에 링크 장애가 발생한다면 문제는 달라질 것이다. 물론 노드의 개수가 작거나 초기에 링크 장애가 발견된다면 처음부터 키 협약을 다시해도 오버헤드가 크지 않겠지만, 노드의 수가 많거나 마지막 라운드에서 링크 장애가 발생한다면 처음부터 다시 하는데 걸리는 시간은 길어질 것이다. 그래서 기존의 방법처럼 하나의 노드와 키 교환을 하는 것이 아니라, 링크장애가 발생했을 때 패턴을 바꾸어서 그 불안정한 링크를 피해서 다른 경로를 통해 키 교환을 하게 된다.

노드의 수를 8개로 확장했을 때의 키 협약을 살펴보면 (그림 6)에서와 같이 8개의 노드가 그룹 키 협약을 통해 키 값을 공유하려고 하는데, 라운드3에서 노드 A-E사이에 링크 장애가 발생한 경우이다. 그러면 키 협약은 더 이상 진행되어질 수 없어서, 다른 파트너를 찾거나 처음부터 다시 시작해야 할 것이다. 이 때 라운드 3에서 노드 A가 그룹 키 값을 얻기 위하여 받아야 할 공개 키 값 $S_{EFGH} = g^{S_{EF}S_{GH}}$는 파트너인 노드 E뿐만이 아니라, 다른 노드 F, G, H도 같은 값을 가지고 있기 때문에 패턴을 바꾸어서 다른 파트너와의 키 교환을 통하여 노드 E와 키 교환을 했을 때와의 동일한 결과 값을 얻을 수 있다.



(그림 6) 라운드 3에서 노드 A와 키 교환을 할 수 있는 노드들

이 때 기존의 기본 패턴은 라운드 3에서 노드 A-E, B-F, D-H, C-G간에 키 협약을 수행해 나가는 것이기 때문에, A-E사이에 링크장애가 발생했을 때 A와 E는 다른 노드들에게 패턴이 바뀌어야 함을 알리고, 모든 노드들은 패턴을 바꾸어서 키 협약을 계속해서 수행해 나가야 할 것이다. 이 방법의 가장 큰 장점은 그룹 키 협약 프로토콜의 효율성에 대한 평가기준인 라운드 수, 키를 위해 교환되어지는 메시지 수의 증가 없이도 하나의 링크 장애를 극복할 수 있다는 장점을 가진다.(<표 1> 참고) 하지만, 패턴의 변화에 대해서는 그룹 내의 다른 참가자에게 반드시 전달이 되어야 한다.



(그림 7) 라운드 3에서 키 교환을 할 수 있는 여러 가지 패턴

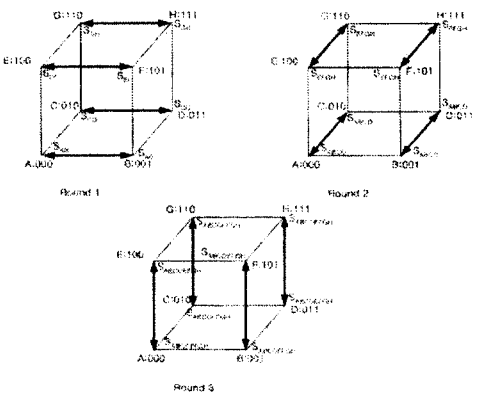
(그림 7)는 8개의 노드가 라운드 3에서 하이퍼큐브 프로토콜을 행하는 여러 가지 패턴이다. 라운드 3에서 가질 수 있는 최대한의 패턴 수는 16개($_4C_1 \times {}_4C_1 = 16$)이다. 이 패턴의 수는 라운드의 번호가 증가할수록 증가한다. 여기서는 4가지의 패턴만 표현하였다.

링크 장애가 발생했을 때 링크 장애에 해당하는 노드는 링크 장애에 대한 사실을 다른 노드들에게 알리고, 메시지를 받은 모든 노드들은 패턴을 바꾸게 된다. 예를 들면, A-E 사이에 링크 장애가 발생하면 A-E는 라운드 3에서 다른 노드들에게 메시지들을 보내고, 모든 노드들은 패턴 1에서 패턴 2 또는 다른 패턴으로 변형해서 키 협약을 수행하게 된다.

3.2 링크 장애를 독립적으로 보완하는 알고리즘

여기에서 제안하는 알고리즘은 3.1절에서의 다양한 패턴을 사용하는 방법이 아닌 하이퍼큐브 프로토콜이 병렬로 키 교환을 한다는 성질을 이용하여 1개의 링크 장애를 모든 노드가 독립적으로 해결하는 방법이다.

(그림 8)은 일반적으로 어떤 문제도 없이 8개의 노드가 하이퍼큐브 프로토콜을 사용해서 키 협약을 수행하는 방법과 각 라운드에서 얻어지는 키 값을 보여준다. 예를 들어 노드 A가 노드 B와의 키 교환을 통해 얻는 키는 라운드 1에서 S_{AB}이고, 라운드 2에서는 노드 C와의 키 교환을 통하여 S_{ABCD}를 얻고, 라운드 3에서는 노드 E와의 키 교환을 통해서 $S_{ABCDEFGH}$를 얻는다. 이런 결과로 (그림 8)에서와 같이 라운드 3에서 모든 노드는 노드 A와 마찬가지로 $S_{ABCDEFGH}$를 가지게 된다. 이 때 각 노드는 디피-헬만 키 교환 방법을 사용하여 키를 공유하게 된다.



(그림 8) 키 협정(노드=8)

이런 키들은 링크 장애가 발생했을 경우 정상적으로 얻어질 수 없게 된다. 하이퍼큐브 프로토콜을 수행하는 두 개의 노드 사이에 링크 장애가 발생했을 때 장애를 인식하는 것은 그 링크에 해당하는 두 개의 노드뿐이다. 다른 노드들은 장애가 발생한 사실을 알 수가 없다. 그렇기 때문에, 장애 발생을 아는 노드들은 그것을 그룹 내에 있

는 다른 노드들에게도 알려주어야 한다. 그래서 앞 절에서 설명했던 방법은 링크장애가 발생했음을 다른 노드들에게 알려주는 절차가 필요하게 된다. 패턴이 바뀌었음을 알아야만 올바른 파트너와 키 협약을 수행할 수 있기 때문이다. 그러나 이 절에서 제안하는 알고리즘은 다른 노드가 링크 장애에 대하여 인식하지 못 할 지라도, 독립적으로 키 협약을 수행해 나갈 수 있는 방법이고 다음과 같이 수행한다.

(1) 라운드 1에서 각 참가자들은 2개의 다른 노드와 디피-헬만 키 교환을 수행한다.

(2) 라운드 2부터는 원래의 하이퍼큐브 프로토콜과 같은 방법으로 키 협약을 행한다. 이 때, 해당 라운드에서 키 교환을 하면서 파트너로부터 받았던 공개 키 값을 저장한다.

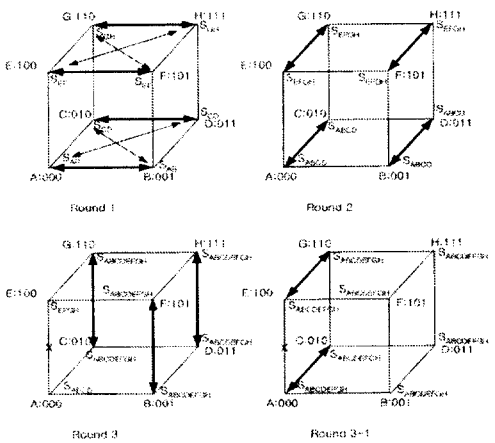
(3) 만약에 노드 N_x가 라운드 i에서 링크 장애가 발생해서 키 협약을 수행할 수 없게 된다면, 라운드 i-1에서 키 교환을 했던 파트너 N_y와 라운드 i에서 다시 키 협약을 수행한다. 이 때 노드 N_x는 새로운 파트너 N_y가 라운드 i에서 받았던 공개 키를 가지고 키 협약을 수행하게 된다.

이 프로토콜에서는 위의 (3)에서 설명한 것처럼 라운드 i에서 링크 장애가 발생할 경우에 바로 이전의 라운드 i-1에서의 자신의 파트너였던 노드를 새로운 파트너로 정하게 된다. 이 알고리즘에서 링크 장애에 관해서 보완하는 방법은 기본적으로 위의 (3)에서 설명한 것처럼, 라운드 i에서 링크 장애가 발생했을 때 링크가 깨진 노드 N_{x1}, N_{y1}들은 새로운 파트너로서 라운드 i-1에서 공개 키를 교환했던 파트너 N_{x2}, N_{y2}를 선택하게 된다.

이 시점에서 N_{x1}, N_{y1}을 제외한 다른 노드들은 라운드 i에서 이미 키 교환을 끝낸 상태이다. 새로운 파트너가 키 교환을 끝낸 상태라는 것은 새로운 파트너 N_{x2}, N_{y2}가 라운드 i에서 자신의 파트너에게서 라운드 i에서의 키 교환을 하기 위한 공개 키를 받았음을 의미한다. 이 공개 키 값은 링크 장애가 발생한 노드가 키 교환을 하기 위하여 받아야만 했던 공개 키 값과 동일하다.

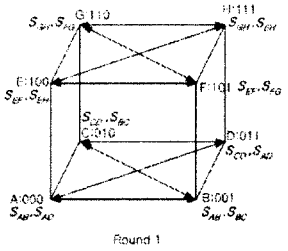
예를 들어 (그림 8)에서 라운드 3에서 노드 A-E 사이에 링크 장애가 발생한다고 하면, 노드 A는 라운드 3에서의 키 교환을 수행할 수 없기 때문에, 라운드 3이 종료된 시점에서 새로운 파트너를 선택하게 된다(그림 9). 새로운 파트너로는 라운드 2에서 노드 A 자신의 파트너였던 C를 선택하게 된다. 여기에서 C는 그룹 키를 얻기 위하여 G에게서 공개 키 S_{EFGH}을 받아서 디피 헬만 키 교환을 사용하여 그룹 키 값인 $S_{ABCDEFGH}$를 가진 상태이다. 이 공개 키 S_{EFGH}은 노드 A가 그룹 키를 얻기 위하여 노드 E에게서 얻어야만 했던 공개 키와 동일하다. 그래서 노드 A는 노드 E와의 키 교환을 통하여, 그룹 키 $S_{ABCDEFGH}$를 가지게 된다.

여기에서 노드 A가 요구하는 공개 키와 노드 A의 이전 라운드의 파트너인 C가 받은 공개 키가 항상 같은 이유는 하이퍼큐브 프로토콜의 특성 상 병렬로 키를 교환하기 때문에 키 교환을 했던 두 개의 노드는 다음 라운드에서 항상 같은 공개 키를 얻는 성질이 있기 때문이다. 예를 들면 라운드 3에서 노드 A와 노드 C는 라운드 2에서의 파트너로서 키 교환 결과 두 노드 모두 키 S_{ABCD}를 가지고 있고, 그리고 라운드 3에서는 두 노드 모두 각자의 파트너에게 공개 키 S_{EFGH}를 받아서 키 값을 공유하게 된다. 이렇게 진행되고 있는 라운드에서의 노드들은 이전 라운드에서의 파트너들과 같은 공개 키를 받아서 키 교환을 수행하게 된다.



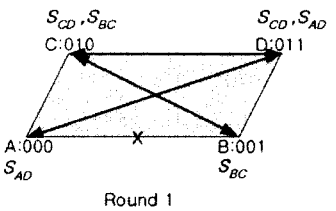
(그림 9) 라운드 3에서 노드 A-E사이에 링크 장애 발생 시 키 교환

위의 과정들은 라운드 1에서 링크 장애가 발생할 경우에는 이전 라운드가 존재하지 않기 때문에 적용되어질 수 없다. 그래서 각 노드들은 라운드 1에서 두 개의 경로를 가지고 다른 두 개의 노드들과 동시에 키 교환을 행해야 한다. (그림 10)은 노드 8개의 그룹 키 협정에서 라운드 1의 키 교환의 예제이다. 노드 A는 노드 B, 노드 D와 동시에 키 교환을 행하여 두 개의 S_{AB}, S_{AD}를 가지게 된다. 다른 노드들도 동일한 방법으로 키를 교환한다.



(그림 10) 노드 8개의 그룹 키 협약에서 라운드 1에서의 키 교환

만약에 노드 A와 B 사이에 링크 장애가 발생한다면 키 교환을 통해 라운드 1에서 노드 A는 S_{AD}를, B는 S_{BC}를, C는 S_{CD}와 S_{BC}를, D는 S_{AD}와 S_{BC}를 가지게 된다. 그래서 라운드 2에서 노드 A와 C는 키 S_{AD}, S_{BC}를 가지고 키 교환을 하고, B와 C는 키 S_{BC}, S_{AD}를 가지고 키 교환을 행하여 모두들 라운드 2에서 $S_{ABCD} = g^{S_{AD}S_{BC}}$를 가지게 된다(그림 11).



(그림 11) 라운드 1에서 노드 A-B사이에 링크 장애가 발생했을 때 키 교환

이 알고리즘에서의 파트너를 찾는 알고리즘은 (그림 12)와 같다.

```
/* 각 round에 대한 노드 알고리즘. */
procedure do_round(round_number)
   mask := mask 00...01   //mask 값을 초기화 한다.
   mask := mask << round_number-1
     //left shift 연산을 한다.
   partner := self_address ⊕ mask  //xor 연산을 한다.
   new_mask := mask >> 1      //right shift 연산을 한다.
   two_party_exchange(partner, new_mask)
end

/* 파트너를 찾고 two-party 교환을 수행하는 재귀함수 */
procedure two_party_exchange(candidate, mask)
   if (mask <= 0)   // leaf 노드에 도달했음을 알려 준다.
     if (link failure 발생)
        mask := mask 00...01 //mask값을 초기화한다.
        mask := mask << round_number-2
        candidate := self_address ⊕ mask
        mask := mask >> 1
        attempt running the two-party DH key exchange
        return success or failure
   endif

   // leaf 노드에 도달하지 않았다면, mask값을 바꿔 준다
   new_mask := mask >> 1  // 오른쪽으로 shift 연산
   result := two_party_exchange(candidate, new_mask)
   if (result = success) return success;

   // tree에서 왼쪽 노드가 실패할 경우, 오른쪽 노드에 시도
   new_candidate := candidate ⊕ mask
   return two_party_exchange(new_candidate, new_mask)
end
```

(그림 12) 링크 장애가 발생했을 때 파트너를 찾는 알고리즘

(그림 12)에서 나타내고 있는 링크 장애가 발생했을 때 파트너를 찾는 알고리즘은 Asokan, Ginzboorg[1]가 제안한 장애 노드에 대하여 파트너를 찾는 알고리즘도 포함하고 있다. 위의 알고리즘에서 mask값이 0일 때, 디피-헬만(DH) 키 교환을 수행하게 되는데, 이 때 DH 키 교환을 수행하는 노드가 장애 노드면 알고리즘에 따라 mask값과 candidate값을 변화시키면서 DH 키 교환을 수행할 새로운 파트너를 찾게 되고, 만약에 링크 장애가 발생했음을 알게 된다면 마스크 값을 초기화시키고 이전 라운드에서의 파트너였던 노드와 키 교환을 수행하게 된다.

4. 알고리즘의 효율성 평가

여기에서는 앞에서 소개했던 하이퍼큐브 프로토콜과 이 논문에서 제시한 링크 장애 극복 하이퍼큐브 프로토콜들을 비교한다. 아래의 표에서 첫 번째는 아무런 장애가 없을 때의 경우를 나타내고, 두 번째는 3.1절에서 설명한 다양한 패턴을 이용하여 링크 장애를 보완하는 방법이고, 세 번째는 3.2절에서 설명한 알고리즘을 따라 링크 장애를 보완하는 방법이다.

<표 1>에서 보이는 바와 같이 이 논문에서 제시한 두 가지 방법들은 링크 장애를 극복하면서도, 기존의 하이퍼큐브 프로토콜에 비교하여, 큰 오버헤드가 없음을 볼 수 있다

특히 3.2절에서 설명한 알고리즘은 1 라운드에서 두 개의 노드와 키 교환을 해서 생기는 메시지 교환의 추가와 링크 장애가 발생했을 경우에 생기는 추가적인 라운드 한 번의 오버헤드만이 있을 뿐이다. 그리고 패턴을 이용하는 방법의 장점 중의 하나는 단지 패턴이 바뀌었음을 알리는 메시지를 추가적으로 필요하기 때문에, 키를 계산하는 오버헤드는 없다는 것이다.

<표 1> 하이퍼큐브 프로토콜과 확장된 방법들의 비교

	Hypercube protocol	제안 알고리즘 1 (패턴을 이용하는 방법)	제안알고리즘 2 (독립적 방법)
exchanges	$\frac{nd}{2}$	$\frac{nd}{2}$	$\frac{n(d+1)}{2}+2$
messages	nd	nd	$n(d+1)+4$
rounds	d	d	$d+1$
synchronous rounds	d	d	$d+1$
특기사항 -link 장애시		링크 장애를 알리는 $2(n-1)$ 메시지 또는 하나의 브로드캐스트 메시지가 필요	

5. 결론

인프라를 제공하지 않는 무선 네트워크인 Ad-Hoc 네트워크에서는 그룹간의 통신을 위한 보안 채널이 필요하다. 이런 보안 채널을 구성하기 위해 메시지의 암호화, 암호해독이 필요하며, 암호화, 암호해독을 위한 키를 요구하게 된다. 이런 그룹 키 협약에 대한 많은 프로토콜들이 제안되어져 왔다.

그러나 이런 많은 프로토콜들은 무선 네트워크에서 발생할 수 있는 링크 장애가 일어났을 경우에 어떻게 보완하는지에 대하여는 다루지 않았다. 그래서 이 논문에서는 하이퍼큐브 프로토콜을 확장하여 많은 오버헤드 없이 링크 장애를 보완하는 2가지 방법을 제안하였다. 첫 번째 방법은 하이퍼큐브 프로토콜에서 패턴을 변형하는 방법으로 링크 장애를 극복 하였고, 두 번째 방법은 하이퍼큐브 프로토콜에서의 구성원간 병렬로 키 교환을 하는 성질을 이용해서 링크 장애를 보완하였다. 이 방법들은 오버헤드를 가장 최소화하도록 만들어졌고, 실제로 첫 번째 방법은 장애를 고려하지 않았던 방법에 비해 1개의 브로드캐스트 메시지나 $2(n-1)$개의 메시지가 필요하고, 두 번째 방법은 장애가 발생했을 때 약간의 오버헤드만을 필요로 한다. 여기서 첫 번째 방법과 두 번째 방법을 직접 비교할 수 없는 것이 두 번째 방법은 첫 번째 방법(1개의 브로드캐스트 메시지를 쓸 경우)에 비해 오버헤드가 더 크지만, 노드들이 독립적으로 수행할 수 있다는 장점을 가지고 있기 때문이다.

그룹 키 협약을 위하여 더욱 연구해야 할 과제들은 아직도 많이 있다. 하이퍼큐브 프로토콜을 사용했을 때, 노드의 개수가 2^d이 아니라면 어떻게 할 것인가 하는 문제와 프로토콜의 종류에 따라 그룹 구성원들의 추가, 삭제, 권한의 문제가 발생했을 때 키 협약을 얼마나 효율적으로 처리할 것인지에 대한 것과 리더의 선출 및 프로토콜 진행순서 등에 관한 연구도 진행되어야 한다.

참 고 문 헌

[1] N. Asokan and Philip Ginzboorg. "Key-agreement in ad-hoc networks". Elsevier Preprint, 2000. To appear.

[2] Ingemar Ingemarsson, Donald T.Tang, and C.K. Wong, "A conference key distribution system," IEEE Transactions on Information Theory, vol.IT-28, no.5, pp. 714-720, Sept. 1982

[3] Mike Burmester and Yvo Desmedt, "A secure and efficient conference key distribution system", in Advances in Cryptology - EUROCRYPT '94, May 1994, pp.275-286

[4] Klaus Becker and Uta Wille, "Communication complexity of group key distribution", In Proc. 5th ACM Conference on Computer and Communications Security, pages 1-6, San Francisco, CA USA, November 1998. ACM press.

[5] Michael Steiner, Gene Tsudik, and Michal Waidner, "Diffie-Hellman key distribution extended to group key agreement", In 3rd ACM Conference on Computer and Communications Security, pages 31-37, New Delhi, India, March 1996. ACM Press

[6] Maarit Hietalahti, "Key Establishment in Ad-hoc Network", Helsinki University of Technology, Department of Computer Science and Engineering, 2001

[7] Eric Ricardo Anton, Otto Carlos Muniz Bandeira Durate, "Group Key Establishment in Wireless Ad Hoc Networks," in Workshop on Quality of Service and Mobility, 2002.

[8] Yongdae Kim, Adrian Perrig, Gene Tsudik, "Simple and Fault-Tolerant Key agreement for Dynamic Collaborative Groups", ACM Conference on Computer and Communications Security, 2000

[9] rfc2631, Diffie-Hellman Key Agreement Method, June 1999

[10] William Stallings, "Network Security Essentials: Applications and Standards", 2000

[11] Whitfield Diffie, Martin E. Hellman, "New Directions in Cryptography", IEEE Transactions on Information Theory, 1976

[12] C.K Toh, "Ad Hoc Mobile Wireless networks : Protocols

이 영 준

1988 고려대학교 전산과학과 (이학사)
1994 미국 미네소타대학교 (전산학 Ph.D.)
현재 한국교원대학교 컴퓨터교육과 교수
관심분야: 정보통신, 지능형 시스템, 컴퓨터교육
E-Mail: yjlee@.knue.ac.kr

민 성 기

1988 고려대학교 전산과학과 (이학사)
1989 Dept. of computer science, University of London 석사
1994 Department. of computer science, University of London [QMW] 박사
현재: 고려대학교 컴퓨터학과 교수
관심 분야: 무선인터넷, 이동통신
E-mail: sgmin@korea.ac.kr

이 성 준

2002 고려대학교 컴퓨터학과 학사
2004 고려대학교 대학원 컴퓨터학과 석사
현재 LG 전자 DM연구소 연구원
관심 분야: 무선인터넷, 이동통신
E-mail: juninjx@lge.com