이동 애드혹 네트워크에서 세션 키 설정 방안
(A Session Key Establishment Scheme in Mobile Ad-Hoc Networks)

왕 기 철 † 정 병 호 †† 조 기 환 †††
(Gi-Cheol Wang) (Byung-Ho Chung) (Gi-Hwan Cho)

요 약 이동 애드혹 네트워크는 무선채널을 통해 데이타가 전송되고 중앙의 관리기관이 존재하지 않으므로 가용자원이 빈약하고 여러 가지 보안위협에 노출되어 있다. 따라서 임의의 두 노드간에 안전한 통신을 수행하기 위해서는 안전하고 통신부하가 작은 세션 키 설정 방법이 요구된다. 그러나 애드혹 네트워크를 위한 기존의 키 설정 방법들은 모든 노드들에게 동일한 그룹 키를 분배하거나 효율적인 공개키 관리를 위한 방법들에 집중되어 왔다. 본 논문에서는 각 노드들간에 안전하게 그리고 작은 부하로 세션 키를 설정하는 방법이 제안된다. 제안하는 방법은 비밀 분산기법과 Diffie-Hellman 키교환 방법을 이용한다. 클러스터내의 안전한 통신을 위해, 각 노드들은 클러스터내에서는 부분 비밀 키를 사용하여 인증을 수행한 후에 자신의 클러스터 헤드와 세션 키를 생성한다. 또한 다른 클러스터간의 안전한 통신을 위해서 각 노드들은 상대방 노드의 공개 키와 Diffie-Hellman 키교환 방법을 이용하여 세션 키를 설정한다. 제안하는 방법은 실험을 통하여 클러스터 헤드 인증기반의 방법[1]에 비해 우수하다는 것을 입증하였다.
키워드 : 애드혹 네트워크, 세션 키 설정

Abstract Mobile Ad-Hoc network tends to expose scarce computing resources and various security threats because all traffics are carried in air along with no central management authority. To provide secure communication and save communication overhead, a scheme is inevitable to securely establish session keys. However, most of key establishment methods for Ad-Hoc network focus on the distribution of a group key to all hosts and/or the efficient public key management. In this paper, a secure and efficient scheme is proposed to establish a session key between two Ad-Hoc nodes. The proposed scheme makes use of the secret sharing mechanism and the Diffie-Hellman key exchange method. For secure intra-cluster communication, each member node establishes session keys with its clusterhead, after mutual authentication using the secret shares. For inter-cluster communication, each node establishes session keys with its correspondent node using the public key and Diffie-Hellman key exchange method. The simulation results prove that the proposed scheme is more secure and efficient than that of the Clusterhead Authentication Based Method[1].
Key words : Ad-Hoc network, session key establishment

1. 서 론

이동 애드혹 네트워크는 공유된 무선 채널을 통해 AP나 기지국과 같은 유선 기반구조의 도움 없이 편리한 통신을 가능하게 해준다[1,2]. 따라서 애드혹 네트워크에서 각 노드는 라우터로서 동작하여야 한다. 이러한 종류의 네트워크는 전쟁터에서의 통신, 재난구조 상황에서의 통신, 그리고 교실이나 회의실에서 즉석통신과 같은 다양한 형태의 응용을 실현할 것으로 보인다.

이동 애드혹 네트워크가 널리 사용되기 위해서는 네트워크 상의 각 노드들에게 안전한 통신을 보장할 수 있어야 한다. 무선 네트워크는 공기 중에 방송되는 전송 특성으로 인해 여러 가지 보안상 위협에 취약한 면을 지니고 있다[1]. 이러한 위협들의 예로는 무선채널을 통한 엿듣기, 트래픽 분석과 같은 수동적인 공격과 악의적인 사용자로부터의 서비스 거부 공격(Denial Of Service), 재전송(replay)공격, spoofing공격과 같은 능동적인 공격이 있다. 그러므로 애드혹 네트워크는 이러한 공

† 비 회 원 : 전북대학교 컴퓨터통계정보학과
gcwang@dcs.chonbuk.ac.kr
†† 비 회 원 : 한국전자통신연구원 무선랜보안연구팀 연구원
cbh@etri.re.kr
††† 종신회원 : 전북대학교 전자정보공학부 교수
ghcho@dcs.chonbuk.ac.kr
논문접수 : 2003년 11월 19일
심사완료 : 2004년 4월 21일

격에 대한 안전한 통신을 보장하기 위해서 기밀성, 인증, 무결성, 부인봉쇄, 그리고 가용성과 같은 보안서비스를 제공하여야 한다. 이러한 보안상의 요구는 적절한 키 관리 방법을 필요로 한다.

일반적으로 유선 네트워크 상에서의 키 관리는 인증기관(Certification Authority)이나 신뢰할만한 키 분배 서버(Key Distribution Server)를 통하여 이루어진다. 일반 노드들은 그들의 서비스를 받아 키를 안전하게 분배 받아 사용한다. 따라서 그러나 애드혹 네트워크는 신뢰할 만한 제삼의 기관이 존재하지 않으므로, 네트워크에 참여한 노드들끼리 협력적으로 키의 분배 및 관리를 수행하여야 한다. 특히, 애드혹 네트워크에서는 모든 노드가 공격 대상이 되므로, 공격에 대한 탐지 및 대처가 어렵다.

애드혹 네트워크에서의 키 관리를 위한 기존의 연구들은 네트워크내의 모든 노드가 동일한 그룹 키를 설정하기 위한 연구나 인증기관의 기능을 각 노드에게 분산시켜 효율적으로 공개키 관리를 수행하기 위한 연구들이 대부분이다. 그러나 애드혹 네트워크에서 그룹 키를 사용하는 경우에, 임의의 두 통신 주체간의 경로는 다중 홉으로 구성되어 있기 때문에 통신경로상의 중간 노드들은 모든 데이타 패킷을 엿듣는 것이 가능하다. 또한 애드혹 네트워크에서 공개키 관리를 위한 인증기관의 기능을 분산시켜 운영하는 경우에, 인증서의 분배, 수정, 폐기, 갱신에 따른 많은 양의 통신 오버헤드를 유발한다.

이에 본 논문에서는 임의의 애드혹 네트워크에서 안전하게 그리고 작은 통신부하로 두 통신 주체간에 세션 키를 설정하는 방법을 제안한다. 제안하는 방법은 검증가능한 비밀 분산기법(verifiable secret sharing)을 이용하여 클러스터헤드와 멤버 노드들간에 상호인증 및 키분배를 수행한다. 이때 생성된 키는 각 노드가 자신의 클러스터내에 다른 노드와 안전하게 통신하도록 해준다. 만일 임의의 노드가 다른 클러스터에 속한 노드와 키를 일치시키기 위해서는 Diffie-Hellman 키교환에 기반하여 자신의 Diffie-Hellman값을 상대방의 공개키로 암호화하고 자신의 클러스터헤드를 소스로 그리고 상대방의 클러스터헤드를 목적노드로 하는 캡슐화 헤더를 붙인다. 캡슐화된 패킷은 자신의 클러스터헤드와 상대방의 클러스터헤드사이의 설정된 경로를 따라 전달된다. 마지막으로 상대방의 클러스터헤드는 캡슐화 헤더를 제거하고 암호화된 패킷을 상대방 노드에게 전달한다. 상대방 또한 앞의 과정을 역으로 수행하여 두 노드간에 세션 키가 설정된다.

본 논문의 구성은 다음과 같다. 2장에서는 애드혹 네트워크에서 대칭 키 관리 및 비밀 분산에 관한 기존의 연구들을 간략히 기술한다. 3장에서는 본 논문에서 제안하는 Diffie-Hellman 및 비밀 분산기법 기반의 세션 키 분배 방법에 대해 기술한다. 4장에서는 제안된 방법과 다른 두 가지 방법의 효율성과 안전성 대한 분석을 하고 5장에서는 결론을 내린다.

2. 관련 연구

2.1 비밀 분산 기법(Secret Sharing)

비밀 분산기법은 몇몇 사용자들에게 부분 비밀 키를 분배하여 단일 사용자의 부분 비밀 키로는 원래의 비밀 키를 복원할 수 없도록 하는 방법이다. (k, n) 비밀 분산 기법[3,4]은 전체 n개의 부분 비밀 키중에서 최소 k개를 병합하여 원래의 비밀 키를 복원할 수 있는 기법이다. (k, n) 비밀 분산 기법은 다음과 같이 동작한다. 비밀 키 S는 n개의 부분 비밀 키 소유자(id_1, id_2, ..., id_n)들에게 분할된다. 이 비밀 키의 생성과 분할은 신뢰된 임의의 dealer의 역할이다.

1. 임의의 소수 p를 선택한다. 이때 p>max(S,n)
2. 임의의 다항식 $f(x) = a_{k-1}x^{k-1} + a_{k-2}x^{k-2} + \cdots + \pmod p$를 생성한다.
   이때 a_0=s이고 a_i (i=1, ..., k-1)은 Z_p로부터 랜덤하게 선택된 값이다.
3. n개의 부분 키(S_i, i=1,2, ..., n)들은 $S_i = f(id_i) \pmod p$에 의해 생성된다.
4. 부분 키들은 각각의 부분 키 소유자들에게 분배된다.

원래의 비밀 키를 복원하기 위해서 Lagrange 보간법을 사용한다. Lagrange 보간법은 다음과 같이 기술된다.

$f(x) = \sum_{i=1}^{k} S_i \times l_{id_i}(x) \pmod p$ 이고, $l_{id_i}(x) = \prod_{i=1, j\neq i}^{k} \frac{x - id_j}{id_i - id_j}$

이다.

그러나 만일 충분히 많은 수의 악의적인 노드들이 부정확한 부분 비밀 키를 전송한다면, 원래의 비밀키 복원은 불가능하게 될 수 있다. 따라서 수신한 부분 비밀 키의 정확성을 검증할 방법이 필요하다[5]. 이러한 기법을 검증가능한 비밀 분산기법이라 한다. 수신된 부분 비밀 키들을 검증하기 위한 방법은 다음과 같이 동작한다.

1. dealer는 부분 비밀 키들을 분배하기 전에 공유하는 다항식의 공계수에 대한 증거인 $g^{a_0}, g^{a_1}, \cdots, g^{a_{k-1}}$을 공표한다.
2. 각 노드는 다른 노드로의 부분 비밀 키를 수신할 때에 다음의 계산을 수행하여 검증한다.

$$g^{S_i} = g^{a_0} \cdot (g^{a_1})^{id_i} \cdot \ldots \cdot (g^{a_{k-1}})^{id_i^{k-1}} = g^{a_0 + a_1 id_i + \ldots + a_{k-1} id_i^{k-1}}$$

그러나 위의 방법은 수신한 부분 비밀키값의 정확성만을 검증할 수 있다. 즉, 송신자가 실제 부분 비밀키값을 소유하고 있는지의 여부는 확인할 수 없다. 참고문헌

[5]에서는 부분 비밀 키 송신자의 부분 비밀키 소유여부를 확인하기 위한 프로토콜이 제안되었다.

2.2 분산 인증기관 (Distributed CA)기법

(k, n) 부분 분산 인증기관 기법[6-8]은 인증기관의 기능을 특정 노드 집합에게 분배하는 방법이다. 이러한 특정 노드들은 각각 자신이 가진 시스템 부분 비밀 키를 이용하여 부분 인증서를 발행할 수 있다. 만일 임의의 노드가 자신의 인증서를 요구하게 되면, 이러한 특정 노드들이 각각 자신의 부분 비밀키로 서명된 부분 인증서를 그 노드에게 전송한다. 그 클라이이언트 노드는 이러한 부분 인증서 k 개를 병합함으로써 자신의 유효한 인증서를 획득하게 된다.

부분 분산 인증기관 기법[7,8]에서 인증기관의 기능을 수행하는 노드들의 수는 제한되어 있고 분산되어 있으므로 각 클라이언트는 자신의 인증서를 얻기 위해 많은 통신 오버헤드를 유발한다. 인증서를 갱신하거나 취소하는 경우에도, 이 방법은 또한 인증서의 동기화를 위한 추가적인 통신이 필요하다.

반면에, 완전 분산 인증기관 기법[6]은 네트워크내의 모든 노드들이 인증기관의 기능을 수행하도록 하는 기법이다. 따라서 이 방법은 인증서를 요청하는 노드들에게 높은 가용성을 제공하고 네트워크 전체에서 발생하는 트래픽의 양을 감소시킨다. 그러나 이 방법은 모든 노드가 부분 비밀 키를 보유하고 있으므로, 일부의 특정 노드만 부분 비밀 키를 가진 부분 분산 기법에 비해 많은 부분 비밀 키들이 위험에 노출된다. 따라서 이 기법에서 임계치 파라미터 k값은 공격자가 짧은 시간 내에 k개의 부분 비밀 키를 얻을 수 없도록 커야한다. 그러나 k값이 커지면 그에 따라 가용성에 다시 영향을 미치게 된다.

두 가지 분산 인증기관 기법은 네트워크가 분할되었다가 다시 병합되는 경우 동기화 문제를 일으킨다. 예를 들어, 하나의 네트워크가 두 네트워크로 분할되었을 때, 각각 부분 키를 수정했다고 가정해보자. 이후에 다시 두 네트워크가 합쳐져 하나의 네트워크로 된다면, 이때 인증기관의 기능을 수행하는 노드들은 서로 다른 비밀 키의 부분 키 값들을 보유하게 될 것이다.

2.3 클러스터 구조 기반의 그룹 키 관리 기법

일반적으로 애드혹 네트워크에서 클러스터 구조는 클러스터 헤드들이 다수의 멤버 노드들과 직접 연결되게 한다. 따라서 클러스터 헤드들은 자신들이 가진 정보를 멤버들에게 즉시 전달할 수 있다. 따라서 클러스터 구조에서 각 클러스터 헤드는 클러스터를 대표하는 지역 방송자 역할을 수행할 수 있다[2].

참고문헌 [9]에서는 애드혹 네트워크에서 모든 노드들이 같은 그룹 키를 공유하는 방법을 제안되었다. 이 방법은 먼저 클러스터를 구성하고 몇몇 클러스터 헤드들을 임시 키 관리자(Potential Key Manager)들로 선정한다. 임시 키 관리자들은 그들의 키들을 생성하고 모든 클러스터 헤드들에 전파한다. 각 클러스터 헤드들은 그 키들 중에서 하나만을 임의의 기준에 따라 그룹 키로 선정한다. 그룹 키가 일치된 후에, 각 클러스터 헤드는 그 그룹 키를 자신의 멤버들에게 전송한다. 그러나 이 기법은 단지 임의의 노드에 대한 개별인증이 아닌 그룹에 대한 멤버여부에 대해서만 인증을 수행한다. 더구나 초기 그룹 키로부터 대부분의 임시 키들이 유도되므로 , 그룹 키가 노출되면 전체 시스템의 보안이 크게 파괴된다.

2.4 클러스터 헤드 인증 기반의 세션 키 설정 방법

참고문헌 [1]에서는 클러스터 헤드가 자신의 멤버노드들을 대신하여 인증을 수행한 후에, 두 통신주체사이에 세션키를 분배하고 이 세션키를 이용하여 메시지를 인증하는 방법이 제안되었다. 제안된 방법에서는 새로 네트워크에 진입하는 노드나 클러스터 사이를 이동하는 노드들의 인증을 위해 시스템 공개키/비밀키 쌍을 이용한다. 이 키쌍은 모든 노드들이 네트워크에 진입하기 전에 미리 분배받는다. 또한 각 노드는 클러스터 내에서 안전한 통신을 수행하기 위해서 클러스터 키를 사용한다. 이 클러스터 키는 각 클러스터 헤드들이 생성해서 자신의 멤버노드 들에게 분배한다. 다른 클러스터에 속하는 노드들간에 세션키를 설정하기 위해서, 각 클러스터 헤드들은 자신의 공개키/비밀키 쌍을 이용하여 상대방 클러스터헤드를 인증한다. 이를 위해 각 클러스터 헤드는 자신의 공개키를 네트워크 전역적인 방송을 통하여 모든 다른 노드들에게 알린다. 그림 1은 클러스터 헤드 인증기반 방법에서 이용되는 키의 구조를 보여준다.

그림 1에서 노드 A와 B가 세션 키를 설정하는 경우에, 노드 A의 클러스터 헤드 3과 노드 B의 클러스터 헤드 1은 다음과 같은 과정을 통해 세션 키를 설정한다.

1. 클러스터 헤드 3은 클러스터헤드 1의 공개 키를 자신의 비밀 키로 서명하여 클러스터 헤드 1에게 전송한다.
2. 클러스터헤드 1은 클러스터헤드 3의 공개키로 복호화하여 자신의 공개키와 일치하는지 검사한다. 만일 일치하면 클러스터 헤드 1은 세션 키를 생성하고, 클러스터 헤드 3의 공개 키와 세션 키를 자신의 비밀 키로 서명하여 클러스터 헤드 3에게 전송한다.
3. 클러스터 헤드 3은 클러스터 헤드 1의 공개키로 수신된 메시지를 복호화하고 자신의 공개 키와 일치하는지 검사한다. 만일 일치하면 클러스터 헤드 3은 수신된 세션 키를 자신의 클러스터 키를 이용하여 멤버노드 A에게 전송한다. 이후에 클러스터 헤드 3은 헬로 메시지를 세션 키로 암호화하여 전송한다.

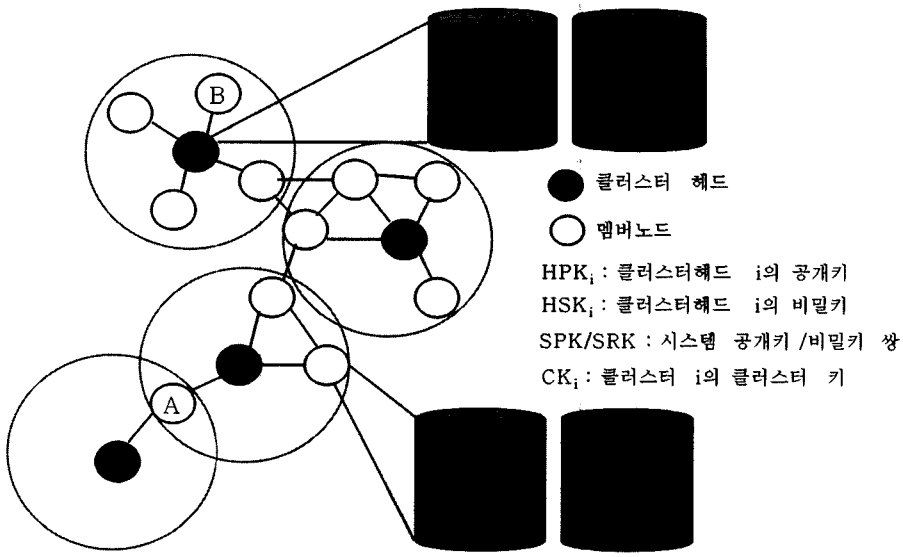


그림 1 클러스터 헤드 인증기반의 세션 키 설정 방법에서의 키 구조

4. 클러스터 헤드 1은 수신된 헬로 메시지를 세션 키를 이용하여 확인하고 노드 B에게 세션 키를 전송한 후에 세션키 설정을 종료한다.

이 방법은 클러스터 헤드들이 자신의 헤드 공개 키를 모든 클러스터 헤드에게 분배해야 하므로 통신 오버헤드가 크다. 또한 통신 상대방이 어떤 클러스터에 속하는지 알아야 하기 때문에 이에 따른 추가적인 통신 오버헤드를 유발한다. 더구나 세션 키의 분배는 반드시 클러스터 헤드들을 통해 이루어지므로, 클러스터 헤드들은 통신 주체인 것처럼 위장할 수 있다. 또한 클러스터내의 통신은 모두 같은 키를 사용하므로 세션 키가 클러스터내의 모든 노드들에게 노출될 수 있다.

3. 클러스터 구조 및 비밀 분산기법에 기반한 세션 키 설정

본 논문에서 제안하는 방법은 클러스터내의 키 설정과 클러스터간의 키 설정 두 가지로 나뉜다. 클러스터내에서의 키 설정은 클러스터 구조가 생성되거나 유지될 때, 각 클러스터 헤드와 각 멤버는 서로를 인증한 후 Diffie-Hellman키 교환에 기반 하여 둘 사이의 고유한 세션 키를 생성한다. 이때 생성된 세션 키들은 클러스터내의 통신 시 각 멤버와 클러스터헤드와의 통신에만 이용된다. 이후 각 노드의 공개 키와 클러스터 헤드는 라우팅 정보의 교환 시에 첨부되어 교환된다.

만일 임의의 노드가 다른 클러스터내의 노드와 키 설정을 원한다면, Diffie-Hellman 키 교환에 기반 하여 두 노드간에 세션 키를 생성한다. 이미 알려진 것처럼, Diffie-Hellman 키 교환 방법은 위장공격 혹은 man-in-the-middle공격에 취약하다. 애드혹 네트워크는 무선을 사용하고 이동을 전제로 하기 때문에 이러한 공격들이 유선에서의 그것들에 비해 매우 쉽다. 이를 해결하기 위해 제안하는 방법은 키 설정을 시도하는 통신주체를 숨기는 접근법을 취한다. 즉 키 설정 과정 동안 키 설정을 시도하는 통신주체가 상대방노드의 공개키로 자신의 Diffie-Hellman값을 암호화 시키고 자신의 클러스터헤드를 소스노드로 하고 상대방노드의 클러스터헤드를 목적노드로 하는 캡슐화 헤더를 붙여 자신의 클러스터 헤드에게 전송한다. 클러스터 헤드는 이 키 설정 패킷을 라우팅한다. 올바른 수신자만이 암호화된 키 설정 패킷을 복호화할수 있으므로, 수신측의 클러스터 헤드는 수신한 패킷을 자신의 멤버 노드들에게 방송한다. 최종적으로 키 설정 패킷을 수신하는 노드는 위와 같은 과정을 역 경로를 통해 수행한다.

제안하는 방법은 다음과 같은 가정들에 기반한다.

- 부분 키 분배 서버는 다항식 $f(x) = a_{k-1}x^{k-1} + a_{k-2}x^{k-2} + \cdots + \pmod p$를 생성한다.
- 각 노드들은 임의의 곱셈그룹 Z_p^*의 생성자 g를 알고 있다.
- 각 노드들은 다항식의 공계수에 대한 증거인 g^{a_0}, $g^{a_1}, \cdots, g^{a_{k-1}}$을 부분 키 분배 서버로부터 수신한다.
- 각 노드들은 애드혹 네트워크에 진입하기 전에 부분 키 분배 서버로부터 자신의 부분비밀키들($S_i = f(id_i) \pmod p$)을 out-of-band방식으로 수신한다. 이러한 부분 비밀 키들은 각 노드들이 클러스터내의 키 설정시 자신을 인증받기 위한 값이다. 또한 각 노드들

은 자신의 공개키 공표시에 자신의 것임을 증명하기 위해서 부분 비밀 키들을 사용한다.

- 각 노드는 애드혹 네트워크에 진입한 후에 자신의 공개 키/비밀 키 쌍을 만들고 저장한다.
- 각 노드들은 동일한 해쉬함수 h()를 사용한다.

3.1 클러스터내에서의 키 설정

클러스터내의 키 설정은 클러스터 헤드와 각 멤버가 안전하게 통신할 수 있는 세션 키를 생성하는 과정이다. 애드혹 네트워크내에서 클러스터 구조는 노드들의 이동으로 인하여 동적으로 변화된다. 클러스터 구조가 변하면 노드들의 역할 또한 변경되므로, 클러스터내의 키 설정 또한 대응되게 변화되어야 한다. 이는 키 설정을 위한 메시지교환을 증가시키므로 네트워크에 큰 부담이 된다. 따라서 제안하는 방법에서는 클러스터 구조의 유지를 위한 메시지에 키 설정을 위한 메시지를 첨부하여 전달하도록 한다.

제안하는 방법은 먼저 클러스터 헤드와 각 멤버가 서로에 대한 인증을 수행한 후에 세션 키를 생성한다. 세션 키의 생성은 Diffie-Hellman 키 교환 프로토콜에 기반한다. 인증 시에 각 노드들은 자신이 보유한 부분 비밀 키를 이용하여 자신의 존재를 입증한다. 이때 인증 프로토콜은 DLEQ[5] 프로토콜을 애드혹 네트워크의 특성에 맞게 변형시킨 것이다. 표 1은 제안하는 클러스터내의 키 설정 프로토콜에서 사용되는 표기법들을 기술한 것이다.

표 1 클러스터내 키설정 프로토콜에서의 표기법과 그 의미들

표기법	의미
$f(id_i)$	노드 i의 미리 분배된 부분키
W_i	노드 i의 난수, $f(id_i)$와 함께 챌린지 값을 생성하기 위해 사용
R_i	노드 i의 난수, 세션키 설정을 위한 Diffie-Hellman값 생성에 사용
r_i	노드 i가 상대방에게 자신의 부분키 소유 사실을 인증받기 위해 사용

제안하는 클러스터 헤드와 멤버 노드들간의 인증 및 키 설정 프로토콜은 다음과 같다.

1. 클러스터 헤드로 선정된 노드는 임의의 난수 2개 (w_{CH}, R_{CH})를 생성하고 클러스터 헤드선언 메시지에 $g^{f(id_{CH})}, a_{CH} = g^{w_{CH}}, g^{R_{CH}}$를 첨부하여 방송한다. 여기서 $g^{f(id_{CH})}$와 a_{CH}는 클러스터 헤드가 인증 받기 위해 미리 건네주는 정보이고 $g^{R_{CH}}$는 키 설정을 위한 Diffie-Hellman값이다.
2. 메시지를 수신하는 각 멤버들은 자신의 난수들(w_i,R_i)을 생성하고 챌린지 값 $c = h(g^{f(id_{CH})}, a_{CH})$, $g^{f(id_i)}, a_i = g^{w_i}$를 포함하는 메시지를 클러스터 헤드에게 전송한다. 여기서 $g^{f(id_i)}$와 a_i는 각 멤버가 인증 받기 위해 클러스터헤드에게 미리 건네주는 정보이다.
3. 클러스터헤드는 $r_{CH} = w_{CH} - f(id_{CH})c \pmod p$와 멤버들에 대한 챌린지 값 $c_i = h(g^{f(id_i)}, a_i)$를 계산한다. 클러스터 헤드는 r_{CH}와 모든 c_i값들을 방송한다.
4. 메시지를 수신하는 각 멤버들은 $a'_{CH} = g^{r_{CH}} \cdot g^{f(id_{CH})c}$를 계산하고 $h(g^{f(id_{CH})}, a'_{CH}) = c$인지 검사한다. 만일 그렇다면 각 멤버들은 $r_i = w_i - f(id_i)c_i \pmod p$를 계산하고 클러스터 가입 메시지에 r_i와 g^{Ri}를 첨부하여 클러스터 헤드에게 전송한다. 이때 g^{Ri}는 키 설정을 위한 Diffie-Hellman값이다. 그렇지 않은 경우에 키 설정은 실패로 끝난다. 또한 $(g^{R_{CH}})^{Ri}$를 계산하고 클러스터헤드와의 세션 키로 사용한다.
5. 클러스터 가입메시지를 수신하는 클러스터헤드는 $a'_i = g^{r_i} \cdot g^{f(id_i)c_i}$를 계산하고, $h(g^{f(id_i)}, a'_i) = c_i$인지 검사한다. 만일 그렇다면 $(g^{Ri})^{R_{CH}}$를 계산하고 그 멤버노드와의 세션 키로 사용한다.

위의 과정을 통해 생성된 세션 키들은 클러스터내에서 각 멤버간에 통신 혹은 클러스터헤드와 멤버간의 통신에 사용된다. 만일 임의의 노드들의 이동으로 인하여 클러스터 구조가 변경된다면, 변경된 영역에 속한 노드들은 위의 프로토콜을 수행 하여 클러스터내의 키 설정을 만들어 낸다.

3.2 클러스터간의 키 설정

임의의 노드가 다른 클러스터에 속한 노드와 안전하게 통신하기를 원할 때에 제안하는 방법은 상대방의 공개 키와 클러스터 헤드를 이용하여 키 설정을 만들어 낸다. 이때의 키 설정 또한 Diffie-Hellman 키 교환 프로토콜에 기반 한다. 모든 노드들의 공개 키와 클러스터 헤드는 라우팅 메시지의 교환 시에 전달되므로, 각 노드들은 이러한 정보를 쉽게 획득할 수 있다. 이때 각 노드 i는 자신이 공개 키의 소유주임을 증명하기 위하여 자신의 공개 키와 함께 $g^{f(id_i)}$를 전송하고 수신자는 2.3절에서 기술한 검증방법에 의하여 수신된 공개 키의 소유주를 검증한다. 이때 각 노드는 다른 노드의 부분 비밀 키를 알 수 없으므로, 자신이 악의적으로 만든 공개 키를 다른 노드의 공개키 인 것처럼 위장할 수 없다. 또한

라우팅 메시지에 대한 안전성 제공은 참고문헌 [10]을 비롯한 많은 문헌들에서 다루고 있으며 본 논문에서는 다루지 않는다. 제안하는 클러스터간의 키 설정 프로토콜은 다음과 같다. 편의상 노드 A가 다른 클러스터의 노드 B와 키 설정을 원한다고 가정한다.

1. 임의의 노드 A는 난수($R_A \in Z_p^*$)를 생성하고, 키 설정을 위해 공개되는 Diffie-Hellman값(g^{R_A})을 계산한다. 계산한 값을 포함하는 패킷을 노드 B의 공개키로 암호화하고 새로운 캡슐화 헤더를 붙인다. 이 캡슐화 헤더의 소스는 노드 A의 클러스터 헤드이고 목적지는 노드 B의 클러스터 헤드이다. 노드 A는 자신의 클러스터 헤드에게 캡슐화된 패킷을 전송한다.
2. 노드 A의 클러스터헤드는 이 캡슐화된 패킷을 설정된 경로에 따라 포워딩한다.
3. 노드 B의 클러스터헤드는 캡슐화 헤더를 제거하고 암호화된 키 설정 패킷을 방송한다.
4. 각 멤버 노드들은 자신의 비밀 키를 이용하여 키 설정 패킷을 복호화 한다. 의도된 수신자 B만이 노드 A의 Diffie-Hellman 값(g^{R_A})을 알 수 있다.
5. 노드 B를 소스로 하고 노드 A를 목적지로 한 뒤 1부터 4의 과정을 수행한다.

제안하는 방법에서 클러스터 헤드들은 키 설정을 원하는 노드들을 대신하여 키 설정 패킷을 중계하지만, 자신이 키 설정의 주체인 것처럼 위장할 수 없다. 이는 클러스터 헤드들이 키 설정의 실질적인 주체를 알 수 없기 때문이다.

제안된 클러스터간 키 설정 프로토콜의 이해를 돕기 위해 도식적인 예제를 도입해보자. 그림 2는 임의의

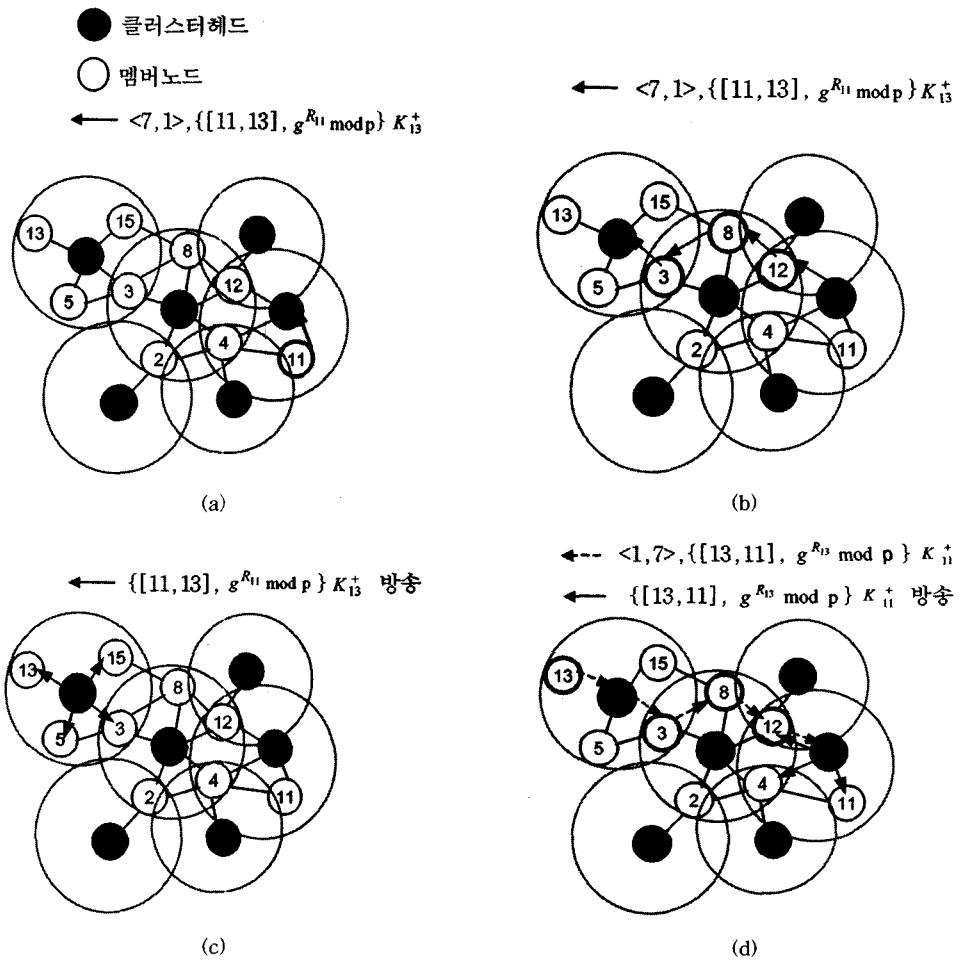


그림 2 클러스터간 키 설정 프로토콜의 예

표 2 클러스터간 키설정 프로토콜에서의 표기법들과 그 의미들

표기법	의미
<S,D>	캡슐화 헤더의 소스노드와 목적노드
[S,D]	IP헤더의 소스노드와 목적노드
K_A^+	노드 A의 공개키
{d} K_A^+	노드 A의 공개키로 데이타 d를 암호화

애드혹 네트워크에서 제안된 프로토콜의 동작을 보여준다. 그림 2에서 사용된 표기법들의 의미는 표 2와 같다.

그림 2에서 노드 11은 노드 13과의 세션 키를 설정하고자 한다. 그림 2의 (a)에서 노드 11은 먼저 임의의 난수(R_{11})를 생성하고 자신의 Diffie-Hellman값($g^{R_{11}}$)을 계산한다. 노드 11은 노드 13에게 계산된 Diffie-Hellman값을 패킷에 포함시키고 이 패킷 전체를 노드 13의 공개키로 암호화 시킨다. 이후 노드 11은 자신의 클러스터 헤드(7)를 소스로 하고 노드 13의 클러스터 헤드(1)를 목적노드로 하는 캡슐화 헤더를 붙여서 클러스터 헤드 7에게 전송한다.

클러스터헤드 7과 1사이의 경로상에 있는 노드들은 캡슐화된 패킷을 포워딩한다(b). 클러스터 헤드 1은 캡슐화된 헤더를 제거하고 암호화된 키 설정 패킷을 자신의 멤버들에게 방송한다. 의도된 목적지 노드 13은 자신의 비밀키로 패킷을 복호화하고 세션 키($(g^{R_{11}})^{R_{13}}$)를 생성한다(c).

이후 노드 13은 노드 11과 같은 절차를 통해서 자신의 키 설정 패킷을 노드 11에게 전송한다(d). 의도된 목적지 노드 11은 자신의 비밀키로 패킷을 복호화하고 세션 키($(g^{R_{13}})^{R_{11}}$)를 생성한다.

위에서 언급된 클러스터간 키 생성 프로토콜은 on-demand 방식으로 동작한다.

3.3 제안된 방안에 관한 논의

본 논문에서 제안하는 세션 키 설정 방법은 각 노드들이 네트워크를 구성한 후에 각각 공개 키/비밀 키 쌍을 만들어 저장하는 것을 가정한다. 그러나 이렇게 생성된 공개 키를 이용하지 않고 새롭게 세션 키를 생성하는 이유는 실제 응용에서의 데이타를 공개 키로 암호화하고 비밀 키로 복호화 하는 연산비용과 시간낭비가 너무 크기 때문이다. 실제로 유선 네트워크에서도 이러한 이유로 인하여, 데이타에 대한 암호화 및 복호화는 대칭키로 하고, 이 대칭 키의 암호화 및 복호화만 공개 키와 비밀 키를 이용한다.

본 논문에서 제안하는 방법은 임의의 두 노드사이에 교환되는 키 설정 패킷의 소스와 목적노드 그리고 Diffie-Hellman값만을 공개키로 암호화하고 비밀키로 복호화하도록 함으로써 세션 키를 설정하게 된다. 따라서 각 노드가 가지는 공개 키와 비밀키는 임의의 두 노드간에 세션 키를 생성하기 위해서만 사용되고, 설정된 세션 키는 각 노드간에 안전하게 데이타를 주고받기 위한 암호화 및 복호화 키로 사용된다.

제안된 방법의 클러스터내의 키 설정에서 임의의 악의적인 공격자는 무선의 방송특성으로 인해 전송되는 공개값들을 쉽게 얻을 수 있다. 그러나 클러스터헤드나 멤버노드가 그 공개값들을 공표하기 위해 사용한 난수들은 이산대수의 어려움으로 인하여 쉽게 획득할 수 없다. 즉 악의적인 공격자는 각 노드가 미리 분배받은 부분키값과 그 노드가 생성한 난수값을 알지 못한다면, 상대방의 챌린지값에 대한 정확한 응답을 생성할 수 없다. 따라서 악의적인 노드가 노드들의 부분키값과 난수들을 알지 못하는 동안에는 클러스터 내의 키설정 프로토콜은 안전하게 동작한다.

제안된 방법의 클러스터간 키 설정은 여러 중간노드들을 거쳐 이루어지게 된다. 따라서 중간노드 중에서 악의적인 노드들은 엿듣기, 재전송공격, DoS(Denial of Serive)공격, 그리고 man-in-the-middle 공격과 같은 다양한 공격을 수행할 수 있다.

먼저 제안된 방법의 엿듣기 공격과 재전송 공격의 공격에 대한 안전성은 이산대수의 어려움에 기초한다. 따라서 이 문제를 해결하지 못하는 한 공격자는 유용한 정보를 얻을 수 없다.

DoS공격은 반드시 키 설정의 문제에만 국한된 공격이 아니므로 비정상 행위의 탐지 및 예방문제[11,12]와 연계되어 해결책을 고안해야 한다.

man-in-the-middle공격의 경우, 제안된 방법은 키 설정의 주체들을 숨기는 방법을 사용하므로 공격자는 공격대상의 설정에 어려움을 겪게 된다. 즉, 공격자가 키 설정 메시지를 엿듣는다고 해도, 공격자는 키설정 주체가 속한 클러스터만을 알 수 있다. 더구나 키 설정 메시지는 수신자 노드의 공개키로 암호화 되므로 그 수신자의 비밀키를 알지 못하는 공격자 노드는 man-in-the-middle공격을 성공할 수 없다. 따라서 임의의 공격자 노드가 키 설정 메시지에 표시되는 클러스터에 속한 모든 노드의 비밀 키를 알지 못하는 한 제안된 클러스터간 키 설정 프로토콜은 안전하게 동작한다.

만일 일정시간 내에 c개의 악의적인 노드들이 어떠한 방법으로 자신의 이웃 노드들에 대한 비밀키를 k개 만큼씩 얻고, 상호 교환하여 가능한 많은 수의 비밀 키를 획득했다고 가정하자. 이때 임의의 악의적인 노드가 수행하는 man-in-the-middle 공격에 대한 제안된 방법의 안전성을 고려해보자. 실제 각 클러스터내의 멤버 수는 클러스터 마다 다르지만 분석의 편의상 l개라고 가정한

다. 위의 상황을 가정할 때 임의의 키 설정 메시지에 포함된 클러스터의 한 멤버노드의 비밀키가 악의적인 노드에게 노출될 확률 p_c는 다음과 같다.

$$p_c = \left(1 - \left(\frac{{}_{n-1-l}C_k}{{}_{n-1}C_k}\right)^c\right)$$

또한 키 설정 메시지에 포함된 클러스터의 모든 멤버 노드의 비밀 키가 임의의 악의적인 노드에게 노출될 확률 p_l은 다음과 같다.

$$p_l = \left(1 - \left(\frac{{}_{n-1-l}C_k}{{}_{n-1}C_k}\right)^c\right)^l$$

4. 실험결과

이장에서는 참고문헌 [1]에서 제안되었던 클러스터 헤드 인증기반의 방법과 제안한 방법을 효율성과 안전성 측면에서 비교한 실험결과를 제공한다. 실험을 위한 시뮬레이터는 Mesquite사에서 개발한 CSIM 18 엔진을 이용하여 개발되었다. 본 시뮬레이터는 일정 영역 내에서 랜덤하게 이동하며 주기적으로 패킷을 교환하는 노드들을 프로세스로 모델링 하였으며 클러스터 구성 및 유지와 세션 키 설정과정을 사건중심기법(Event driven method)에 의해 수행하였다.

실험은 500m×500m 영역에서 45개의 노드를 가지고 600초 동안 수행되었다. 모든 노드들은 초기에 랜덤하게 배치되고 매초마다 랜덤한 방향으로 이동한다고 가정한다. 키 설정은 Hello메시지 주기사이에 한번씩 발생하도록 하였고, 키 설정을 시도하는 노드들은 랜덤하게 선정된다. 실험을 위한 입력 파라미터들은 표 3과 같이 설정된다.

표 3 실험 파라미터들

파라미터	값
노드들의 수	15, 25, 35, 45
노드의 이동속도	A(0~5m/s), B(5~10m/s), C(10~15m/s)
전송범위	20~200meter
정지시간	0, 10, 20초
Hello메시지 간격	3초
키 설정 시도횟수	200회
악의적인 CH의 비율	20%, 30%, 50%

먼저 키 설정 방법의 효율성을 비교하기 위해 키 설정을 위해 전송되는 메시지의 수를 척도로 사용한다. 이때 경로설정 및 유지를 위한 제어메시지는 포함되지 않으며 두 방법이 같은 라우팅 프로토콜을 사용하는 것으로 가정한다. 두 방법들은 노드들의 전송거리와 노드들의 수를 변화시켜 가면서 비교된다.

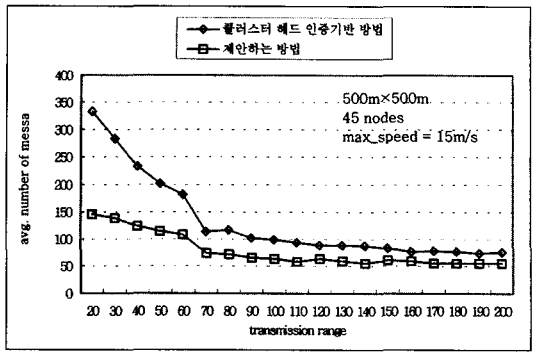


그림 3 키 설정을 위해 전송되는 메시지수 vs. 전송거리

그림 3은 전송거리의 변화에 따른 키 설정을 위해 전송되는 평균 메시지 수의 변화를 보여준다. 전송거리가 증가함에 따라 전송되는 메시지의 수도 감소된다. 이는 전송거리가 증가함에 따라 클러스터의 수가 감소되기 때문이다. 모든 전송거리에서 제안하는 방법은 클러스터 헤드 인증기반 방법에 비해 작은 수의 전송메시지를 유발한다. 이는 제안하는 방법은 네트워크 전역으로 방송되는 메시지가 없기 때문이다. 전체 전송거리에서 제안하는 방법은 클러스터헤드 인증기반 방법에 비해 약 40%정도의 메시지를 감소시킨다.

그림 4는 같은 전송거리(40m, 100m, 150m)에서 노드의 수를 변화시켰을 때의 영향을 보여준다. 전송거리가 짧은 경우(=40m)에는 노드의 수가 증가함에 따라 클러스터의 수도 증가되므로, 키 설정을 위해 전송되는 메시지의 수 또한 증가된다. 그러나 전송거리가 긴 경우(=100m, 150m)에는 노드의 수가 증가되더라도 클러스터의 수가 급격히 증가되지 않는다. 이는 새로 진입하는 노드들의 대부분이 기존의 클러스터에 소속되기 때문이다. 모든 전송거리에서 제안하는 방법은 클러스터헤드 인증기반의 방법보다 작은 전송메시지를 발생시킨다. 특히 낮은 전송거리에서 두 방법의 전송메시지수의 차이는 노드수의 증가에 따라 증가됨을 볼 수 있다.

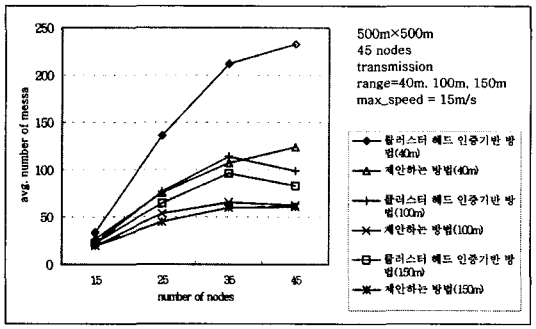


그림 4 키 설정을 위해 전송되는 메시지수 vs. 노드의 수

키 설정 방법의 안전성을 비교하기 위해 man-in-the-middle공격에 대한 견고성 여부를 비교한다. 일반적으로 키 설정 프로토콜에 대한 많은 공격이 가해질 수 있다. 그 중에서 가장 심각한 위협을 초래하는 공격은 man-in-the-middle공격이다. 이는 이 공격들이 세션 키들의 무결성을 훼손시키기 때문이다.

클러스터 헤드 인증기반 방법에서 세션 키 설정 주체의 클러스터 헤드들만 man-in-the-middle공격을 쉽게 수행할 수 있다. 이는 클러스터 헤드들을 통해 키 설정 주체들을 위한 세션 키가 분배되기 때문이다. 또한 제안하는 방법에서 키 설정 주체들의 클러스터 헤드들은 MAC 계층의 패킷 snooping을 통해 키 설정의 소스 혹은 목적노드를 알 수 있으므로 다른 노드들에 비해 man-in-the-middle공격을 수행하기가 용이하다. 즉, 클러스터 헤드가 아닌 노드들은 키 설정의 주체가 숨겨져 있으므로 man-in-the-middle공격의 대상을 설정하기가 매우 어렵다. 따라서 본 실험에서는 키 설정 주체들의 클러스터 헤드들만 man-in-the-middle공격을 수행한다고 가정한다.

이 실험에서 simulate된 악의적인 클러스터헤드들의 행위는 다음과 같다. 악의적인 클러스터헤드는 자신이 소스노드로부터 수신한 대칭 키 요구를 IP spoofing을 이용하여 자신이 요구한 것처럼 위장하여 목적노드에게 전송한다. 목적노드는 악의적인 클러스터헤드와 세션 키를 설정하게 된다. 또한 악의적인 클러스터헤드는 자신이 목적 노드인 것처럼 위장하여 자신의 공개값을 소스노드에게 전송한다. 소스노드는 악의적인 클러스터헤드와 세션 키를 설정하게 된다. 이러한 공격의 효과는 키 설정 패킷 내의 소스노드 혹은 목적노드가 악의적인 클러스터 헤드인 패킷의 수가 증가되는 것이다. 반면에 제안하는 방법에서는 악의적인 클러스터 헤드들이 소스노드 혹은 목적노드로 위장할 수 없다. 물론 악의적인 클러스터 헤드들이 실제 소스 노드이거나 목적노드일수도 있다. 하지만 제안하는 방법에서는 그 외의 키 설정 패킷들을 위조하기는 어렵다. 본 실험에서는 각 방법에 대해 20%, 30%, 그리고 50%의 악의적인 클러스터 헤드들을 가지고 비교하였다.

그림 5는 악의적인 클러스터헤드들의 증가에 따른 위조된 패킷의 수를 보여준다. 이때 위조된 패킷은 소스노드 혹은 목적노드가 악의적인 노드로 위조된 패킷을 말한다. 실험의 편의성을 위해 악의적인 클러스터헤드가 실제 소스노드 혹은 목적노드인 경우도 위조된 패킷의 수에 포함되었다. 악의적인 클러스터헤드의 수가 증가함에 따라 위조된 패킷의 수도 증가한다. 특히 클러스터헤드 인증기반 방법에서는 악의적인 클러스터헤드의 비율이 크게 증가할수록 위조된 패킷의 수도 크게 증가된다.

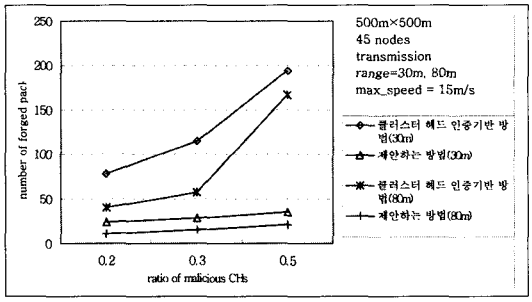


그림 5 위조된 패킷의 수 vs. 악의적인 클러스터헤드의 수

이는 악의적인 클러스터 헤드가 자신의 모든 멤버들에 대한 키 설정 패킷을 위조하기 때문이다. 반면에 제안하는 방법은 악의적인 클러스터헤드가 키 설정 패킷의 소스노드 혹은 목적노드를 알지못하므로 키 설정의 주체에게 위조된 패킷이 도달할 확률이 아주 낮다.

5. 결 론

본 논문에서는 클러스터 구조와 Diffie-Hellman 키 교환 방법에 기반 하여 애드혹 네트워크에서 안전하게 세션 키를 설정하는 방법을 제안하였다. 제안하는 방법은 클러스터내에서는 각 노드들에게 분배된 부분 비밀키를 이용하여 인증을 수행한 후에 키 설정을 수행하고 클러스터간에는 클러스터 헤드와 통신주체의 공개 키를 이용하여 키 설정을 수행한다. 제안하는 방법은 실험결과에 의해 클러스터헤드 인증기반의 키 설정 방법에 비해 전송되는 메시지 수를 40%정도 감소시킨다는 것을 알 수 있었다. 또한 제안하는 방법은 악의적인 클러스터헤드에 의한 위장공격에도 키 설정 패킷의 위조를 어렵게 한다는 것을 알 수 있었다. 따라서 제안하는 방법은 클러스터헤드 인증기반의 세션 키 설정 방법에 비해 효율적이며 안전하다고 할 수 있다.

참 고 문 헌

[1] L. Venkatraman and D.P. Agrawa, "A Novel Authentication scheme for Ad-Hoc Networks," Proc. of IEEE WCNC 2000, 2000, pp. 1268-1273.

[2] C. Chiang, H. Wu, W. Liu and M. Gerla, "Routing in Clustered Multihop, Mobile Wireless Networks with Fading Channel(CGSR)," Proc. of IEEE SICON'97, 1997, pp. 192-211.

[3] Y. Frankel and Y.G. Desmedt, "Parallel Reliable Threshold Multisignature," Univ. of Wisconsin-Milwaukee, Technical Report TR-92-04-02, 1992.

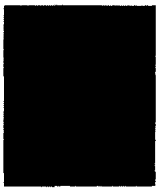
[4] A. Shamir, "How to Share a Secret," Communication of the ACM, 22(11), pp. 612-613, 1979.

[5] B. Schoenmakers, "A Simple Publicly Verifiable Secret Sharing Scheme and its Application to

Electronic Voting," CRYTO '99, LNCS(1666), pp. 148-164, 1999.
[6] J. Kong, P. Zerfos, H. Luo, S. Lu, and L. Zhang, "Providing Robust and Ubiquitous Security Support for Mobile Ad-Hoc Networks," IEEE ICNP 2001, 2001.
[7] S. Yi and R. Kravets, "Key Management for Heterogeneous Ad-Hoc Wireless Networks," Univ. of Illinois, Urbana-Champaign, Technical Report UIUCDCS-R-2002-2290, 2002.
[8] L. Zhou and Z. J. Haas, "Securing Ad-Hoc Networks," IEEE Networks, 13(6), pp. 24-30, 1999.
[9] S. Basagni, K. Herrin, E. Rosti, and D. Bruschi, "Secure Pebblenets," Proc. of the ACM Symposium on MobiHoc, 2001, pp. 156-163.
[10] K. Sanzgiri, B. Dahill, B. Levine, C. Shields, E. Belding-Royer, "A Secure Routing Protocol for Ad-Hoc Networks," Proc. of ICNP'02, 2002, pp. 78-87.
[11] S. Marti, T. J. Giuli, Kevin Lai, and M. Baker, "Mitigating routing misbehavior in mobile ad hoc networks," Proc. of the 6th Int'l Conference on Mobile Computing and Networking, 2000, pp. 255-265.
[12] L. Buttyan and J. P. Hubaux, "Stimulating Cooperation in Self-Organizing Mobile Ad Hoc Networks," ACM/Kluwer Mobile Networks and Applications(MONET), 8(5), pp. 579-592, 2001.
[13] C.R. Blakely, "Safeguarding Cryptographic Keys," Proc. of Nat. Computer Conf, 1979, pp. 313-317.
[14] J-P. Hubaux, L. Buttyan, and S. Capkun, "The Quest for Security in Mobile Ad-Hoc Networks," Proc. of the ACM Symposium on MobiHoc, 2001.
[15] N. Asokan, P. Ginzboorg, "Key Agreement in Ad-hoc Networks," Computer Communication Review, 23(17), pp. 1627-1637, 2000.

왕 기 철
정보과학회논문지 : 정보통신
제 31 권 제 1 호 참조

조 기 환
정보과학회논문지 : 정보통신
제 31 권 제 1 호 참조



정 병 호
1988년 전남대학교 전산통계학과 학사 2000년 충남대학교 컴퓨터과학과 석사 2000년 3월~현재 충남대학교 컴퓨터과학과 박사수료. 1998년 2월~2000년 6월 국방과학연구소 선임연구원. 2000년 6월~현재 한국전자통신연구원 무선LAN 보안연구팀 팀장