이동 Ad Hoc 네트워크를 위한 대칭키 관리 기법

송지은° 왕기철 조기환
전북대학교 컴퓨터 정보 학과
{jeusong°, gcwang, ghcho}@dcs.chonbuk.ac.kr

A Symmetric Key Management Scheme for Mobile Ad Hoc Networks

Jieun Song° Gicheol Wang Gihwan Cho
Dept. of Computer and Information, Chonbuk National University

요 약

Ad Hoc 네트워크는 무선의 고유특성으로 인해 여러 가지 보안상 위협에 취약한 면을 지니고 있다. 이러한 위협들의 예로는 무선채널을 통한 엿듣기, 트래픽 모니터링과 같은 수동적인 공격과 악의적인 사용자로부터의 서비스 거부 공격(Denial Of Service), 그리고 신뢰성이 손상된 개체나 도난 당한 장치로부터의 공격등과 같은 능동적인 공격이 있다. 임의의 네트워크 환경에서 이러한 공격에 대한 안전한 통신을 보장하기 위해서 기밀성, 인증, 무결성 그리고 가용성등을 충족시켜야 한다. 이러한 보안상의 요구는 적절한 키 관리 방법을 필요로 한다. 하지만 기존의 방법들은 키의 일치를 위해 과도한 통신 오버헤드, 오랜 지연시간을 요구하거나 안전상 취약점을 노출한다. 본 논문에서는 호스트가 이동하는 상황에서, 빠르게 비밀키를 공유하도록 클러스터 구성을 이용하고 보다 안전한 키 관리를 위해 임계치 암호화 방식을 사용하는 방법을 제안한다. 더불어 제안하는 방법은 프로토콜에서 사용되는 임시키들과 부분키들을 주기적으로 update하여 안전성을 향상 시키며, 호스트들이 이동하는 상황에서도 안전하게 비밀키를 공유하도록 해준다. 따라서 본 논문에서 제안하는 방법은 이동 Ad Hoc 네트워크에서 높은 가용성을 보장하고 보다 안전하게 그룹키나 세션키를 공유하는 방법으로 이용될수 있다.

1. 서 론

이동 Ad Hoc 네트워크는 무선의 특징으로 인해 여러 가지 보안상 위협에 취약한 면을 지니고 있다. 이러한 위협들의 예로는 무선채널을 통한 엿듣기, 트래픽 모니터링과 같은 수동적인 공격과 악의적인 사용자로부터의 서비스 거부 공격(Denial Of Service), 그리고 신뢰성이 손상된 개체나 도난 당한 장치로부터의 공격등과 같은 능동적인 공격이 있다. 이러한 공격에 대한 안전한 통신을 보장하기 위해서 기밀성, 인증, 무결성, 그리고 가용성등을 충족시켜야 한다. 이러한 보안상의 요구는 적절한 키 관리 방법을 필요로 한다.

기존의 유선 기반의 네트워크에서의 대칭키 관리는 유선의 물리적인 안전성과 희박한 위상변경으로 인해 키의 안전한 분배에 초점을 맞추었다. 그러나 Ad Hoc 네트워크는 특성상 잦은 위상변경과 높은 에러율로 인해 키의 일치(Agreement)가 실패하기 쉽다. 따라서 Ad Hoc 네트워크에서 키의 관리는 키의 가용성과 안전성에 초점을 맞추어야 한다. 하지만 기존의 방법들은 키의 일치를 위해 과도한 통신 오버헤드와 오랜 지연시간을 요구하거나 안전상 취약점을 노출한다.

본 논문에서는 호스트들이 이동하는 환경에서 작은 통신 오버헤드를 통해 안전하게 키의 일치 및 update를 수행하는 키 관리 방법을 기술한다. 제안하는 방법은 초기에 임의의 호스트들에게 비밀공유키의 부분비밀키(secret share)를 할당한다. (n, t)임계치 암호화를 통해 이들 중에서 t개의 부분비밀키를 얻으면 원래의 비밀공유키를 를 환원시킬수 있다. 제안하는 방법은 먼저 네트워크를 클러스터 구조로 구성하고, 이웃 클러스터 헤드들 간의 통신을 통해 t개의 부분비밀키를 획득한다. 그리고 t개의 부분키들을 이용해 원래의 비밀키를 복원한다. 이 과정에서 각각의 부분비밀키가 유효한지 점검하고, 원래의 비밀키를 얻은 다음에는 기존의 부분비밀키를 수정한다. 클러스터 구성을 위한 호스트간의 통신시, 클러스터 헤드들간의 통신시, 클러스터 구성 후 클러스터 내에서의 통신시 각 용도에 따른 임시 대칭키를 이용한다. 이때 클러스터 구성을 위해 사용하는 임시 대칭키와 클러스터 헤드들간의 통신에 사용되는 임시 대칭키들은 모두 각 호스트에 보관된 그룹 인증키로부터 유도된다. 제안하는 방법은 그룹 인증키의 주기적인 변경을 통해 임시 대칭키들과 부분 비밀키들을 계속 변경되게 한다. 따라서 다른 키 관리 방법에 비해 안전하다. 또한 제안하는 방법은 호스트들의 이동에 의한 키 획득의 실패시에도, 안전한 프로토콜을 통해 키를 획득할 수 있도록 해준다.

본 논문의 구성은 다음과 같다. 2장에서는 Ad Hoc 네트워크에서 대칭키 관리 및 임계치 암호화에 관한 기존의 연구들을 간략히 기술한다. 3장에서는 본 논문에서 제안하는 임계치 암호화기반의 대칭키 관리 방법에 대해 기술한다. 4장에서는 기존의 방법과 비교 분석하고 5장에서는 결론을 내린다.

2. 관련 연구

2.1. Secret Sharing

Secret Sharing은 임의의 비밀키를 여러 사용자들이 나누어 갖도록 함으로써 단일 사용자가 자신의 키만으로는 원래의 비밀키를 복원할수 없도록 하는 오래된 방법이다. 이중 가장 대표적인 방법은 전체 n개의 부분키들로 분할되었을 때 이중에서 k개의 부분키를 얻었을 때 비로소 원래의 비밀키를 복원할 수 있는 (k, n) 임계치 암호화 기법[3][4]이다. (k, n) 임계치 기법은 다항식 보간법에 기초하며 다음과 같이 동작한다.

1. 임의의 소수 p를 선택한다. 이때 p>max(S,n)
2. $f(x) = a_{k-1}\, x^{k-1} + a_{k-2}\, x^{k-2} + \ldots + a_1 x + a_0$
   를 임의로 생성한다.
   (이때 a_0=S이고, $a_i, i=1,2,\ldots,k-1$ 는 Z_p로부터 임의로 선택된 값이다.)
3. n개의 부분키(S_i, i=1,2,....,n)들은 $S_i = f(id_i) \pmod p$에 의해 생성된다.
4. 생성된 {소스값,부분키}들은 임의의 n개의 호스트들에게 분배된다.

원래의 비밀키를 복원하기 위해서 Lagrange 보간법을 사용한다. 이때 최소 k개의 부분키들을 얻으면 원래의 다항식 f(x)를 복원 가능하다. 그리고 원래의 비밀키는 f(0)를 계산하는 것으로 얻어진다.

Secret Sharing은 충분한 시간이 주어진다면 t개의 부분키를

얻어 비밀키가 노출될 수있다. 따라서 규칙적으로 분산된 부분키들을 update시킬 필요가 있다[4]. 부분키 update는 다음과 같이 랜덤의 update다항식($f_{update}(x)$)을 생성하여 원래의 다항식 f(x)에 더함으로써 이루어진다. 따라서 임의의 수정된 부분키($S_{i,update}$)는 $f_{new}(id_i)$를 계산하면 얻어진다. 실제로는 $f_{update}(x)$에 각각의 부분키 소유자의 소스값($id_i, i=1,2,...,k-1$)들을 대입하여 새로운 부분키($\hat{S}_i$, i=1,2,...k)를 생성하고 이들을 각각의 부분키 소유자에게 전송해서 update된 부분키를 만들도록 한다. 즉 $S_{i,update} = S_i + \hat{S}_i \pmod p$를 통해 새로운 부분키를 생성한다. 만일 임의의 부분키 소유자가 악의적으로 다른 부분키 소유자들이 원래의 비밀키를 복원하지 못하도록 부분키가 아닌 랜덤값을 전송한다고 가정해보자. 이 경우에, 원래의 비밀키를 복원하리라 기대했던 사용자는 Lagrange 보간법에 의해 전혀 엉뚱한 값(예를 들어 $\hat{S}$)을 추출하게 된다. 이것을 방지하기 위해서는 임의의 부분키 수신시 수신된 부분키가 유효한 값인지 검증할 방법이 필요하다. 이러한 부분키 검증 방법은 다음과 같이 수행된다.

1. 임의의 dealer는 자신의 부분키들을 전송하기에 앞서 임의의 수 g를 선택한다. 이 g에 f(x)의 공계수(a_{k-1}, a_{k-2}, ..., a_1, a_0)들을 멱승하여 $g^{a_{k-1}}, g^{a_{k-2}},..., g^{a_1}, g^{a_0}$ 를 구한다. 이 dealer는 이 값들을 공표한다.
2. 각 호스트는 자신이 임의의 부분키들을 수신할 때, 이 부분키 값의 유효성을 다음과 같이 계산해서 검증한다.

$$(g^{a_{k-1}})^{id_i^{k-1}} \cdot (g^{a_{k-2}})^{id_i^{k-2}},...,(g^{a_1})^{id_i} \cdot g^{a_0}$$
$$= g^{a_{k-1} id_i^{k-1} + a_{k-2} id_i^{k-2} + ... + a_1 id_i + a_0}$$
$$= g^{S_i}$$

2.2. 클러스터 기반의 비밀키 관리 기법

[2]에서는 센서 네트워크에서 주어진 비밀키를 계속해서 update하는 방법을 제안하였다. 이 방법은 2 단계로 구성된다. 첫 번째 단계는 클러스터 구성 단계이다. 이 단계에서는 몇 개의 호스트들을 임의의 weight에 따라 그룹으로 묶는 작업을 수행한다. 이때 그룹으로 묶는 기준들은 배터리 잔량, 다른 호스트와의 거리 등의 파라미터들을 사용한다. 그리고 이들을 가중치와 곱하고 더해서 하나의 weight으로 만들어 낸다. 클러스터 구성시 이웃 클러스터로부터 메시지를 수신하는 호스트는 이 사실을 클러스터 헤드에게 알려 이웃에 다른 클러스터가 존재함을 알린다. 이것은 인접 클러스터 헤드간의 백본 네트워크 형성이라 불린다. 두 번째 단계에서는 먼저 임시 키 관리자를 선출한다. 이때 인접 클러스터 헤드와 비교했을 때 자신의 weight 값이 가장 큰 클러스터 헤드가 임시 키 관리자가 된다. 각 임시 키관리자는 자신의 weight값과 생성한 키를 클러스터 헤드 백본에 방송한다. 이때 중간의 클러스터 헤드들과 멤버 호스트들은 이 값들을 전달하는 중계노드의 역할을 수행한다. 다른 임시 키 관리자의 키 값을 수신하는 임시 키관리자는 자신의 weight값 보다 크면 수신된 값으로 변경하고, 그렇지 않다면 자신의 키 값을 유지한다.

이 방법에서는 모든 호스트들에 대한 인증을 호스트 각각에게 미리 공유된 임의의 그룹키(k_{Gi})를 이용하여 수행한다. 또한 클러스터 선정과정을 위한 헬로 메시지 키(k_h^i), 클러스터 간의 안전한 통신을 위한 클러스터 키(k_c^i), 그리고 클러스터 헤드 백본에서의 통신을 위한 백본 키(k_b^i)를 이용한다. 이때 i값은 키 update를 수행하는 라운드수 이고 초기값은 0이며 키 update를 수행할 때마다 1씩 증가한다. 헬로 메시지 키와 백본 키는 미리 각 호스트에 공유된 해쉬함수(h())를 통해 각각 다음과 같은 방법으로 획득 되어진다.

$k_b^0 = k_b^i = k_{Gi}$, $k_h^i = h(k_b^{i-1}) = h^i(k_{Gi})$, $k_b^i = h(k_h^i) = h^{i+1}(k_{Gi})$

클러스터 기반의 비밀키 관리방법은 각 호스트들에게 공유된 그룹키(k_{Gi})에 지나치게 의존하기 때문에 만일 그룹키가 노출되면 전체 시스템의 보안이 크게 훼손된다. 더구나 이 방법은 센서 네트워크를 위하여 설계 되었기 때문에, 이동을 전제로 하는 이동 Ad Hoc 네트워크로의 응용에는 많은 문제점들을 노출한다. 첫째로, 이동을 전제로 하는 Ad Hoc 네트워크에서 이 방법은 다수의 키 관리자가 발생할 확률이 높다. 이 경우 전체 네트워크에서의 비밀 키 공유는 실패하게 된다. 둘째로, 임시 키관리자의 위치는 각 네트워크에 골고루 분산되어 있을수 있고, 각 임시 키 관리자는 모든 임시 키 관리자의 키값을 수신한 후에야 공유된 비밀키를 설정할 수 있다. 따라서 이동을 전제로 한다면 이러한 비밀키 설정을 위한 시간지연은 길어지게 된다. 셋째로, 임의의 호스트가 클러스터 헤드 선정과정 중에 이동하거나 새로운 노드가 네트워크로의 진입시, 이 호스트는 비밀키를 얻지 못하거나 다음 update 라운드까지 대기하여야 한다.

3. 임계치 암호화에 기반한 비밀대칭키 관리 기법

초기에 네트워크 관리자는 임의의 다항식을 생성한다. 이 다항식을 기반으로 n개의 부분키(S_i, i=1,2,...,n)들을 생성한다 ($S_i = f(id_i) \pmod p$). id_i값과 생성된 부분키들을 임의로 선정한 n개의 호스트들에게 분배한다. 각 호스트들은 임의의 소수 g(1<g<p)와 ($g^{a_{k-1}}, g^{a_{k-2}},..., g^{a_1}, g^{a_0}$)을 공통적으로 소유한다. 또한 해쉬함수 h(), 초기 그룹 공유키(k_{G1})를 공통적으로 소유한다. 이때 g와 g에 공계수를 멱승한 값들은 부분키의 유효성을 검증하기 위하여 사용된다. 해쉬 함수 h()는 헬로키와 백본키를 생성하는 데 사용되며 초기 그룹 공유키(k_{G1})는 초기에 비밀키를 한번 복원하는 데 사용되고 이후에는 라운드마다 다른 그룹 공유키들(k_{G2}, k_{G3},..., k_{Gi}, ...)이 사용된다.

3.1. 제안된 방법의 1단계

제안된 방법의 1단계에서는 비밀 키 복원 및 update를 위한 클러스터 구성을 수행한다. 이때의 클러스터 구성은 헬로 키($k_h^i = h^i(k_{Gi})$)를 이용하여 수행되며, 제안된 방법에서는 연결성(connectivity)기반의 클러스터 구성을 수행하는 것으로 가정한다. 이는 되도록 하나의 클러스터에 많은 호스트들을 포함하기 위해서이다. 따라서 각 호스트들 간에 전송되는 메시지는 다음과 같은 형태가 된다.

$$E_{k_h^i}(c_{id} | id \mid MAC_{k_{Gi}}(c_{id} | id))$$

위의 메시지에서 c_{id}는 호스트 id의 연결성(connectivity)을 의미하고, $MAC_{k_{Gi}}(x)$ 는 임의의 메시지 x를 키 k_{Gi}를 사용해서 메시지 인증코드를 만드는 것을 의미한다. 또한 $E_{k_h^i}$는 키 k_h^i를 사용해서 x를 암호화하는 것을 의미하고, 기호 "|" 는 두 메시지의 결합을 의미한다. 연결성의 기준에 따라 클러스터 헤드가 선정되면 각 클러스터 헤드는 자신의 역할을 메시지로 구성하여 멤버들에게 전송한다. 만일 임의의 멤버 호스트가 하나이상의 클러스터 헤드와 인접하다면, 자신의 클러스터 헤드에게 이 사실을 통보한다. 이때의 메시지 구조는 다음과 같다.

$$E_{k_h^i}(id_{sender} | hop\ count | id_{CH} |$$
$$MAC_{k_{Gi}}(id_{sender} | hop\ count | id_{CH}))$$

여기서 id_{sender} 는 통보자의 id, id_{CH}는 이 호스트가 인접한 클러스터 헤드의 id, 그리고 hop count는 통보자와 인접 CH의 홉수를 의미한다. 이로인해 각 클러스터 헤드는 자신의 주변에 얼마만큼의 클러스터 헤드가 존재하는 지를 파악할 수 있다. 이후에 각 클러스터 헤드는 자신의 멤버들에게 클러스터 키(k_c^i)를 생성하여 전송한다. 각 멤버들은 만일 자신이 부분키 값을 가지고 있다면 자신의 부분키 소스값(id_i)과 부분키 값($S_i = f(id_i) \pmod p$)을 클러스터 키로 암호화 하여 클러스터 헤드에게 전송한다. 이로인해 각 클러스터 헤드들은 자신이 임계치 k개의 부분 키중에서 몇 개가 부족한 지를 알수 있게 된다. 따라서 이 부족분을 2단계에서 이웃 클러스터 헤드들을 통해 획득한다. 또한 수신된 부분키들을 호스트들 각자가 보유하고 있는 $g^{a_{k-1}}, g^{a_{k-2}},..., g^{a_1}, g^{a_0}$ 들을 이용하여 멤버들로부터 수신된 부분키가 유효한지 검증한 후에 보관하거나 혹은 폐기시킨다.

3.2. 제안된 방법의 2단계

제안된 방법의 2단계에서는 먼저 주변의 이웃 클러스터 헤드들과 접촉하여 이웃 클러스터 헤드들이 보유하고 있는 부분

키들을 획득한다. 이 절차는 1단계에서 획득한 이웃 클러스터 헤드로의 경로를 통해 이루어 진다. 이를 위해서는 먼저 안전한 통신을 보장하기 위해 클러스터 백본간 통신키($k_b^{i}=h^{i+1}(k_{G1})$)를 생성해야 한다. 각 클러스터 헤드는 이 백본 키를 이용하여 자신이 보유하고 있는 부분키들을 자신의 이웃 클러스터 헤드들과 교환한다. 이때 메시지의 구조는 다음과 같다.

$$E_{k_b^{i}}(id_{sender}|id_i|f(id_i)|id_j|f(id_j)|...|$$
$$MAC_{k_G}(id_{sender}|id_i|f(id_i)|id_j|f(id_j)|...))$$

이러한 과정의 수행에도 불구하고 여전히 임계치 k개의 부분키를 획득하지 못하는 경우는 지금까지 획득된 이웃 클러스터 헤드의 수를 기준으로 부족분($k-NS_{current}$)을 임의의 이웃 클러스터 헤드에게 요청한다. 이때 이미 저장된 부분키 소스값(id_i)을 같이 첨부하여 이웃 클러스터헤드가 중복된 값을 전송하지 않도록 한다. 즉 이때의 메시지 구조는 다음과 같다.

$$E_{k_b^{i}}(k-NS_{current}|id_i|id_j|...|$$
$$MAC_{k_G}(k-NS_{current}|id_i|id_j|...))$$

만일 이웃 클러스터 헤드로부터의 부분 키가 k개에 미치지 못했다면 다른 이웃 클러스터 헤드에게 부족분을 요청해서 k개를 채울 때 까지 반복한다. 2단계에서도 1단계와 마찬가지로 $g^{a_{k-1}}, g^{a_{k-2}},..., g^{a_1}, g^{a_0}$ 들을 이용하여 수신된 부분키들을 1단계에서 처럼 유효성을 검증한 후 보관한다.

3.3. 제안된 방법의 3단계

제안된 방법의 3단계 에서는 k개의 분분키들을 이용하여 Lagrange 보간법으로 원래의 비밀키를 복원한다. 그리고 원래의 다항식에서 사용되었던 공계수들과 비밀키(a_0)를 다음과 같이 변경시킨다.

$$a'_{k-1}=(g^{a_{k-1}})^{i+1} \bmod p,\ a'_{k-2}=(g^{a_{k-2}})^{i+1} \bmod p,...,$$
$$a'_1=(g^{a_1})^{i+1} \bmod p,\ a'_0=(g^{a_0})^{i+1} \bmod p$$

이렇게 변경된 공계수들은 다음과 같이 새로운 그룹 키값을 생성하는데 이용된다.

$$k_{G_{i+1}}=a'_{k-1} \oplus a'_{k-2} \oplus ... \oplus a'_1 \oplus a'_0$$

각 클러스터 헤드들은 새로 생성된 공계수들과 비밀키(a'_0)를 이용하여 다음과 같이 update된 새로운 다항식을 생성한다.

$$f_{update}(x)=a'_{k-1}x^{k-1}+...+a'_1x+a'_0 \pmod p$$

다항식을 생성한 후에는 클러스터 내의 소스값(id_i)들을 적용한 새로운 부분 키 값($f_{update}(id_i)$)들을 생성한다. 마지막으로 각 클러스터 헤드들은 생성된 비밀키와 새로운 그룹 키를 클러스터 키로 암호화 하여 멤버들에게 방송한다 . 이때 각 멤버들은 $g^{a_{k-1}}, g^{a_{k-2}},..., g^{a_1}, g^{a_0}$ 들을 변경 시킨다. 이들은 다음 라운드의 키 update시 수신된 부분 키의 검증을 위해 필요하다. 클러스터 헤드는 또한 새로운 부분 키들을 기존 부분 키의 소유주들에게 전송한다.

위에서 기술한 3단계의 대칭 키 분배 및 update는 주기적으로 이루어지며, update 간격은 네트워크의 보안 요구 정도에 의해 정해진다. 즉 높은 보안이 요구되는 경우는 update간격이 짧고, 그렇지 않을 경우에는 update 간격이 상대적으로 길어지게 된다.

3.4.이동 혹은 새로운 호스트의 진입에 의해 비밀키를 획득하지 못한 경우

이동 Ad Hoc 네트워크에서 각 호스트들은 이동으로 인해 비밀키를 얻지 못하는 경우가 자주 발생한다. 이런 경우에 본 논문에서는 패스워드 인증 기반의 키교환 프로토콜[1]을 응용하여 키분배를 제공한다. 키를 보유하지 않은 사용자가 키를 가진 사용자에게 자신의 id와 자신이 생성한 랜덤 secret의 g에 대한 멱승을 초기 그룹인증키(k_{G1})로 암호화 하여 전송한다. 이를 수신한 키 보유자는 자신의 랜덤 secret을 멱승하여 임시 공유 비밀키 K를 얻어내고, 자신의 랜덤 secret의 g에 대한 멱승과 자신이 만들어 낸 challenge를 K로 암호화 하여 함께 되돌려 보낸다. 이후 임시 공유키인 K를 이용한 challenge교환에 이어, 요청자는 자신이 정당한 사용자임을 밝히기 위해 g의 공계수에 대한 멱승들($g^{a_{k-1}}, g^{a_{k-2}},..., g^{a_1}, g^{a_0}$)을 XOR연산한다. 이 결과를 키 보유자에게 전송하면 키 보유자는 자신의 계산결과와 동일한지 점검하고, 동일하면 요청자에게 라운드 번호(i)와 초기 비밀키(a_0)를 전송한다. 이 메시지를 수신한 요청자는 라운드 값(i)와 $g^{a_{k-1}}, g^{a_{k-2}},..., g^{a_1}, g^{a_0}$ 들을 이용하여 새로운 그룹 인증키($k_{G_{i+1}}$)와 현재의 비밀키 S를 생성한다.

4. 비교분석

표 1에서는 2장에서 기술한 기존의 대칭 키 관리 기법과 본 논문에서 제안한 기법을 지원하는 보안성의 정도, 수동적인 공격에 대한 보안 지원사항, 그리고 능동적인 공격에 대한 보안 지원 사항에 따라 비교하였다. 기존의 방법들이 인증이나 기밀성과 같은 보안 요구사항들을 지원한다 하더라도, 이들의 프로토콜 자체가 호스트가 이동 하는 환경에 적합하지 않거나 이동을 제한하므로 가용성이 크게 감소된다. 또한 이들은 수동적인 공격과 능동적인 공격에 대해 적절한 해결책을 제시하지 못하고 있다. 반면에 제안하는 방법은 대부분의 보안 요구사항을 지원하며, 이동 Ad Hoc 환경에 맞게 높은 가용성을 지원한다. 더불어 프로토콜에 사용되는 키를 주기적으로 update시키고, 능동적인 공격에 상대적으로 강한 방법들을 이용한다.

표1. 대칭키 관리 기법들에 대한 보안성 비교

비교항목 / 관리 방법	보안 요구사항 지원	수동적인 공격에 대한 보안	능동적인 공격에 대한 보안
패스워드 인증기반의 하이퍼큐브	인증, 기밀성, 낮은 가용성	패스워드 기반	고장허용 프로토콜
클러스터 기반의 키교환 프로토콜	인증, 기밀성, 무결성, 낮은 가용성	초기 그룹키 기반	-
제안된 방법	인증, 기밀성, 무결성, 높은 가용성	주기적으로 변경되는 임시 그룹키 기반	임계치 암호화 부분키 값 검증

5. 결 론

본 논문에서는 이동 Ad Hoc 네트워크에서 안전하면서도 빠르게 비밀키를 획득하고 update하는 방법을 제안하였다. 제안하는 방법은 호스트가 이동하는 상황에서, 안전하게 비밀키를 공유하도록 해준다. 또한 클러스터 구성 도중이나 키 update중에 이동하였을 때에도 키 보유자와 미 보유자간의 키교환 프로토콜을 통해 미 보유자가 안전하게 비밀키를 얻도록 해준다. 따라서 본 논문에서 제안하는 방법은 이동 Ad Hoc 네트워크 응용에서 가용성을 크게 높여주며 보다 안전하게 그룹키나 세션키를 공유하는 방법으로 이용될수 있다.

6. 참고문헌

[1] N. Asokan and P. Ginzboorg, "Key Agreement in Ad-hoc Networks," Computer Communication Review, 23(17), pp. 1627-1637, 2000

[2] S. Basagni, K. Herrin, E. Rosti, and D. Bruschi, "Secure Pebblenets", Proc. of the ACM Symposium on MobiHoc, pp. 156-163, 2001

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

[4] L. Zhou and Z. J. Haas, "Securing Ad Hoc Networks," IEEE Networks, 13(6), pp. 24-30, 1999