이동 애드혹 망을 위한 시공간 방식의 보안 라우팅 프로토콜

정회원 조 인 휘*

Secure Routing with Time-Space Cryptography for Mobile Ad-Hoc Networks

Inwhee Joe* ***Regular Member***

요 약

본 논문에서는 이동 애드혹 망을 위해 시공간 방식에 근거한 보안 라우팅 프로토콜의 설계와 성능을 기술한다. 제안된 시공간 방식은 시간영역에서는 출발지와 목적지 간에 키 배포를 수행하고, 또한 공간영역에서는 경로 상에서의 침입 탐지를 수행하기 때문에 붙여진 이름이다. 데이터 인증을 위해 효율성이 높은 대칭키 방식을 사용하고, 비밀 키는 출발지에서 목적지로 시간차를 이용하여 배포되어 진다. 또한, 단방향의 해쉬체인을 공간측면에서 홉단위로 형성함으로써, 침입자나 포섭된 노드가 라우팅 정보를 조작하는 것을 방지한다. 제안된 라우팅 프로토콜의 성능을 평가하기 위해 ns-2 시뮬레이션을 통하여 같은 조건하에서 기존의 AODV와 비교한 결과, 우리가 제안한 프로토콜이 보안 기능을 제공하면서도 동시에 지연시간과 처리율 측면에서 기존의 AODV와 유사한 성능을 보여주고 있다.

Key Words : Secure Routing, Time-Space Cryptography, Mobile Ad-Hoc Networks, Hash Chain

ABSTRACT

This paper describes the design and performance of a secure routing protocol with time-space cryptography for mobile ad-hoc networks. The proposed time-space scheme works in the time domain for key distribution between source and destination as well as in the space domain for intrusion detection along the route between them. For data authentication, it relies on the symmetric key cryptography due to high efficiency and a secret key is distributed using a time difference from the source to the destination. Also, a one-way hash chain is formed on a hop-by-hop basis to prevent a compromised node or an intruder from manipulating the routing information. In order to evaluate the performance of our routing protocol, we compare it with the existing AODV protocol by simulation under the same conditions. The proposed protocol has been validated using the ns-2 network simulator with wireless and mobility extensions.

Ⅰ. 서 론

이동 애드혹(Ad-Hoc) 네트워크는 기지국 혹은 AP(Access Point)에 의한 중앙 집중화 된 관리나 표준화된 지원 서비스 없이 임시 망을 구성하는 무선 이동 단말들의 집합으로, 그 특성상 임시 구성 망이나 재해, 재난 지역이나 전쟁터와 같이 기존의 기반 시설을 이용할 수 없는 환경에 적용하는 것으로 인식 되어 왔다. 이러한 이유로 이동 애드혹 네트워크에 대한 연구는 주로 군사용이나 대체 네트워크용으로 활용할 수 있는 방향으로 연구가 진행되어 왔다.

* 한양대학교 정보통신학부 부교수 (iwjoe@hanyang.ac.kr)
논문번호 : KICS2006-09-393, 접수일자 : 2006년 9월 25일, 최종논문접수일자 : 2007년 1월 12일

이동 애드혹 망은 고정된 기반 구조를 가지고 있는 것이 아니기 때문에 각 이동 단말들의 상태를 파악하고 있어야 한다. 이동 네트워크 위상은 빠르고도 예측할 수 없게 변하기 때문에 이런 이동 단말들의 이동성문제로 인하여 보안에 상당히 취약한 문제에 직면해 있다. 아직까지 애드혹 네트워크를 위해 제안된 여러 종류의 라우팅 프로토콜에는 충분한 보안에 관한 알고리즘이 제시되지 못한 실정이다.

현재 이동 애드혹 네트워크의 보안 관련 연구 분야는 크게 키 관리 기법과 라우팅 인증 프로토콜 등으로 나눌 수 있다[2]. 키 관리는 이동성이 있는 노드들에게 효율적으로 키를 전달 및 공유해주기 위한 연구와 인증 알고리즘을 써서 이동성이 있는 노드들의 패킷에 대한 인증프로토콜을 설계하는 것이 목표이다. 인증 알고리즘의 추세는 해쉬 알고리즘이 많이 쓰이고 있는데, 해쉬 체인은 웜홀(wormhole) 공격에 강한 면을 보이고 있기 때문이다[4]. 그 뿐만 아니라 기존 라우팅 프로토콜과 비교하여 성능 면에서도 차이가 많이 나지 않기 때문이다[5].

이동 애드혹 네트워크의 보안 요구조건은 다른 통신 네트워크에서 요구되는 것과 동일하다. 그러나 이동 애드혹 네트워크에서는 노드의 신분이 서로에게 불확실한 경우가 많으며 멀티홉 방식에 의해 라우팅을[1] 할 경우 악의적인 중간 노드에 의해 발생될 수 있는 데이터의 무결성 및 기밀성 문제가 존재한다. 특히 매체를 신뢰할 수 없는 상황에서 암호를 사용하므로 암호 키에 크게 의존하게 된다. 따라서 키 사이에 신뢰할 수 있는 관계를 형성하고, 이를 이동 애드혹 네트워크 전반에 분배하는 것이 주요 과제가 된다. 한편으로, 보안 문제가 확실히 해결되다 보면 컴퓨팅 문제가 발생되어 노드와 네트워크 전체에 심각한 부하를 주게 되므로, 이러한 trade-off를 고려한 이동 애드혹 네트워크에 적합한 알고리즘, 키 분배 및 인증 프로토콜 개발이 현실적으로 가장 필요하다[1].

본 논문에서는 이동 애드혹 망을 위해 시공간 방식의 보안 라우팅 프로토콜을 제안한다. 제안하는 시공간 방식은 시간영역에서는 출발지와 목적지 간에 키 배포를 수행하고, 또한 공간영역에서는 경로 상에서의 침입 탐지를 수행한다. 2장에서는 제안하는 시공간 방식에 대한 개념을 먼저 설명한 후에, 시공간 방식을 적용한 S-AODV 라우팅 프로토콜에 대해 기술한다. 3장에서는 ns-2 시뮬레이션을 통해 제안한 프로토콜의 성능을 평가하고 마지막으로 4장에서는 결론을 맺는다.

Ⅱ. 보안 라우팅 프로토콜 (S-AODV)

이 장에서는 제안하는 시공간 방식에 대한 개념을 먼저 설명한 후에, 시공간 방식을 적용한 S-AODV 보안 라우팅 프로토콜에 대해 소개한다.

2.1 인증 처리 과정

그림 1은 출발지 노드와 목적지 노드에서의 인증 방법을 설명한 것이다.

처음 출발지 노드에서 생성된 값을 HMAC이라고 하고 패킷에 값을 추가하여 보낸다. 중간노드의 처리 과정을 거친 패킷은 목적지까지 가게 되는데, 목적지에서는 받은 패킷에서 HMAC을 추출하여 자기가 생성한 HMAC1(Time값으로 키 테이블에서 키를 가져와 생성)과 비교를 하여 값이 같다면 RREP패킷을 만들 준비를 하기 위해 자기의 정보를 담은 HMAC과 Hash Chain을 생성한다. 하지만 틀리다면 RERR 패킷 즉 에러를 발생하여서 다시 처음부터 라우팅을 하게 한다. 인증 처리는 출발지 노드나 목적지 노드 둘 다 같은 값의 생성 및 비교를 통하여 인증을 한다.

그림 2는 중간노드에서의 인증 처리 흐름도 이다. 중간노드에서는 RREQ(Route Request) 패킷을 주고받을 때와 RREP패킷을 주고받을 때의 처리가 각각 다른데 그 이유는 RREQ패킷을 보낼 때는 브로드캐스트를 하기 때문에 각 노드의 인증을 하기가 쉽지 않기 때문이며 RREP패킷을 할 때는 유니캐스트 환경에서 오기 때문에 각각의 인증 처리가 다르다. RREQ패킷을 받은 노드들은 자기의 노드주소와 받

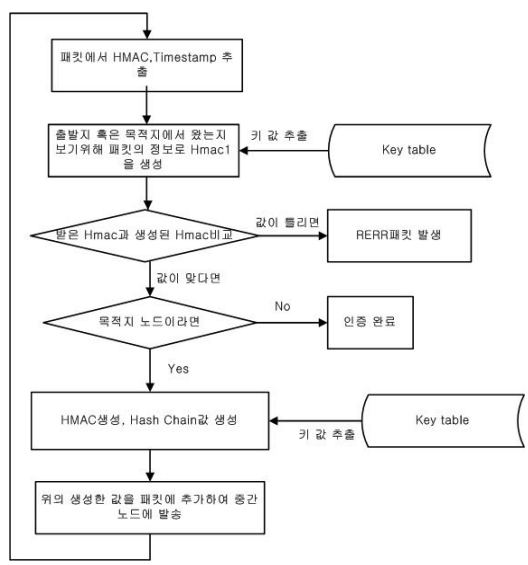

그림 1. 출발지 혹은 목적지 노드에서의 인증 처리 흐름도

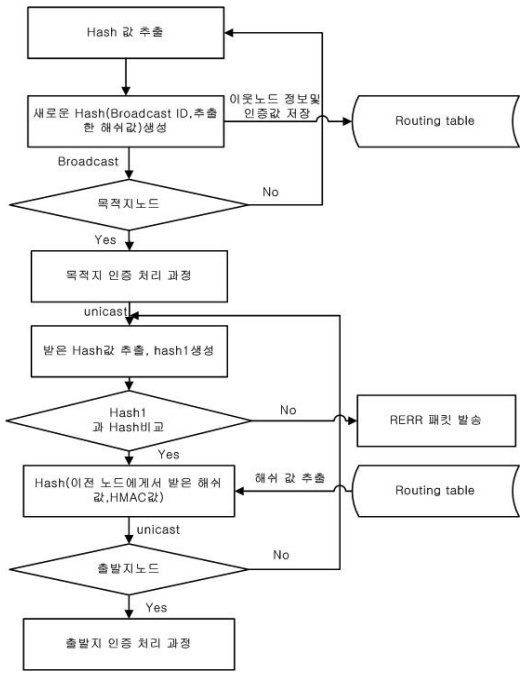

그림 2. 중간 노드에서의 인증 처리 흐름도

은 해쉬만으로 새로운 해쉬값을 만들어 이웃노드들에게 전파하는데 이때 중요한 것은 라우팅 테이블에 해쉬값을 저장한다는 점이다. 이것은 나중에 RREP 패킷이 올 때 자기가 보낸 이웃 노드에게서 왔는지 알 수 있게 해주는 정보를 제공한다.

그리고 목적지의 인증 처리 과정을 거쳐 RREP패킷을 만들어서 중간노드들에게 보냈다면 그 중간 노드들은 자신이 이웃노드에게 보내기 위해 RREQ패킷에 추가한 해쉬값과 HMAC값으로 Hash1을 생성하여 RREP를 보낸 노드에서 온 Hash값이랑 비교한 후 이웃노드에게서 왔다는 것을 인증한다. 그 다음에 RREP패킷을 보낼 때는 다시 RREQ를 보낸 이웃노드의 인증 값(라우팅 테이블에 저장된 값)과 HMAC을 해쉬 화하여 RREQ를 보낸 노드에게 패킷을 보낸다.

다음은 세 개의 노드가 있다고 가정했을 때 출발지 노드, 중간노드, 목적지노드에서의 인증 처리와 사용되는 값에 대한 설명을 하겠다.

Source_HMAC(S,D,key) //HMAC 생성 함수

H_0=Source_SHA(S , HMAC); //해쉬 값 생성 함수

Timestamp= 출발지 노드에서 패킷을 보낸 시간 + delay //시간

그림 3. RREQ일 때 출발지 노드에서의 인증 생성

그림 3은 키 테이블은 모든 노드가 가지고 있고 delay값은 안전한 채널을 통해 전달된다고 가정했을 때, 출발지 노드는 RREQ를 보내기 전에 HMAC값을 만든다. HMAC을 만들 때 쓰는 키는 키 테이블에 있는 값이고, delay값은 출발지 노드나 목적지 노드만이 알 수 있다. 그 키로 출발지 노드에서는 출발지 노드 ID와 목적지 노드ID 그리고 Key로 HMAC값을 생성한다. 그리고 그 값을 보호하기 위하여 출발지 노드ID와 HMAC값을 해쉬화 한다. Timestamp는 RREQ패킷을 보내는 현재 시간에 출발지 노드와 목적지 노드만이 알 수 있는 delay값을 넣어서 보내준다. 이렇게 함으로써 출발지와 목적지 노드 간에 서로만이 알 수 있는 값을 가질 수 있게 된다.

HMAC값과 Timestamp값은 변하지 않는다.

H_1=Middle_SHA(Broadcast ID, H_0)

그림 4. RREQ일 때 중간노드에서의 인증 처리

그림 4는 중간 노드에서는 값을 받고 HMAC값과 Timestamp값은 변경시키지 않는 대신에 오류 검출과 RREP패킷에 쓰기 위하여 Broadcast ID와 받은 해쉬값으로 H_1을 생성한 후 다음 노드에게 패킷을 보낸다. RREQ에서의 처리는 해쉬 체인값만 변할 뿐 다른 값은 변하지 않는다.

```
time=Timestamp-delay  //시간 값
key=키 테이블에서의 키 값 //Time값으로 추출된 키
Destination_HMAC=hash(S,D,key)//출발지에서 보낸 정보로 만든 값에서
if(Destination_HMAC==받은 HMAC값) { //비교
sendReply();// RREP패킷 생성 함수
}else SendError(); //RERR패킷 생성 함수
```

그림 5. RREQ일 때 목적지에서의 인증처리

그림 5는 목적지 노드에게 RREQ패킷이 도착하면 패킷에 있는 HMAC값이 출발지 노드에게서 왔는지 검사를 하기 위해 HMAC을 생성하고 패킷에 있는 값과 비교 후 맞으면 RREP 패킷을 보내고 틀리면 RERR를 보낸다.

Destination e_HMAC(D, Sequence_number ,key)

H_2=Destination_SHA(H_1, HMAC);

Timestamp= Timestamp + delay

그림 6. RREP일 때 목적지에서의 인증 생성

그림 6은 값이 맞는다고 가정할 때 목적지 노드가 sendReply함수를 발생시키면서 자신의 인증을 위하여 자신의 ID와 일련번호 그리고 테이블에서 키를 가져와 HMAC을 생성하고, 해쉬체인을 위하여 이전노드의 해쉬값과 HMAC으로 해쉬값을 생성하여 RREP패킷에 추가한다. 그리고 목적지 인증을 위하여 출발지와 목적지가 가지고 있는 delay로 Timestamp를 변경시킨다.

```
if( Middle_SHA(H1, HMAC) == H2) {
recvReply();//중간에서 처리하기위한 sendReply와 역할이 비슷한 함수
}else SendError(); //에러 처리 함수
H3=Middle_SHA(H0, HMAC); //해쉬 체인값 변경
```

그림 7. RREP일 때 중간노드에서의 인증 처리

그림 7은 중간노드에서는 목적지 노드에게서 온 패킷으로 자기가 보낸 곳에서 왔는지 자신이 생성한 해쉬값과 받은 HMAC값으로 RREP패킷에 있는 해쉬값을 비교한다. 같으면 이전에 받은 해쉬값으로 해쉬값을 생성해서 보낸다. 여기서 목적지에서 RREP를 보낼 때 쓰는 함수랑 recvReply를 할 때 함수가 다른 이유는 출발지와 목적지 그리고 중간 노드에서 사용되는 인증 방법이 다르게 생성되기 때문이다.

```
if( Source_SHA(H0, HMAC) == H3) {
Source_HMAC=hash(D, Sequence_number ,key)//목적지정보로 만든 값
if(Source_HMAC==HMAC) {
printf("인증 및 라우팅 완료");
}else SendError();// 에러 처리
}
}else SendError();//에러 처리
```

그림 8. RREP일 때 출발지에서의 인증 처리

그림 8은 출발지 노드에서는 RREP패킷을 받은 후 목적지 노드에게서 왔는지 확인하기 위해 목적지 노드에서의 방법과 마찬가지로 키 테이블에서 키를 추출한 후에 HMAC값을 생성하여 비교한 후 값이 같으면 인증과 라우팅은 완료가 된다. 그리고 값이 틀리다면 다시 라우팅을 시작한다.

```
S->A: S , D , H1, HMAC_Source, TimeStamp(12:11)
A->B: S , D , H2, HMAC_Source, TimeStamp(12:11)
B->C: S , D , H3, HMAC_Source, TimeStamp(12:11)
C->D: S , D , H4, HMAC_Source, TimeStamp(12:11)
D->C: D , S , H5 , HMAC_Destination, TimeStamp(12:15)
C->B: C , S , H6 , HMAC_Destination, TimeStamp(12:15)
B->A: B , S , H7 , HMAC_Destination, TimeStamp(12:15)
A->S: A , S , H8 , HMAC_Destination, TimeStamp(12:15)
```

그림 9. Route Discovery 예제

그림 9는 S를 출발지 노드 D를 목적지 노드라 가정했을 때 인증에 필요한 값을 위주로 패킷의 내용을 표시하였다.

2.2 패킷 구조와 라우팅 테이블

제안하는 S-AODV에서는 기존 RREQ 패킷에 세 가지 필드를 추가 시킨다. 즉, 보안을 위해 해쉬 체인, HMAC, 그리고 약간 변형된 Timestamp를 넣는다.

또한, 기존 RREP 패킷에 보안을 위해 HMAC (Hash Message Authentication Code), Hash chain 값, 그리고 타임스탬프(Timestamp)를 추가한다. HMAC 값 출발지 노드와 목적지 노드 ID와 key로서 생성되며 해쉬 체인은 이전 노드에서 온 해쉬값과 브로드캐스트 주소로 이루어진다. 하지만 처음 출발지 노드에서는 해쉬값이 출발지 노드 ID와 HMAC값으로 이루 어진 해쉬값이 들어간다.

Hash Chain에서 이전 노드의 해쉬값과 브로드 캐스트 주소로 해쉬값을 만드는 이유는 S->A, A->B.., 로 보안상의 위험성이 감지될 경우 각 2개의 쌍으로 묶어 에러 찾기에 유리하게 하기 위함이다.

Timestamp는 키를 결정할 때 사용된다. HMAC 기본적으로 비밀키가 필요하다. 그러므로 S->D까지의 라우팅에 노드들은 비밀키를 가지고 있어야 하는데, 그 키는 키 관리 테이블에서 관리하고, 각 시간 구간별로 정해진 키를 사용하여 인증을 확인하는데 사용이 된다. 예를 들어 S->A에서 패킷에 보내진 시간을 보면 MAC값에 사용된 비밀키를 알게 되므로 인증을 할 수 있다. 그리고 Timestamp에는 한 가지 트릭이 있는데 delay값을 줌으로써, 공격자는 키 테이블을 가지고 있더라도 delay값이 없다면 제대로 된 키 값을 가져올 수 없다.

예를 들면 Timestamp값이 0:0:5(시:분:초)이라고 할 때 0:0:12(delay time)를 더해 줌으로써 Timestamp값은 0:0:12가 되고, 아무것도 모르는 공격자는 12의 값으로 키 테이블의 값을 가져오지만 해쉬화 할 때 다른 값이 생성됨으로 소스나 혹은 목적지에서는 인증 처리에서 값이 틀리다는 것을 알 수 있다. delay 값을 아는 목적지나 소스는 12값을 빼버림으로써 5라는 값을 추출해서 올바른 키 값을 가져올 수 있다.

AODV가 기존 라우팅 알고리즘과 다른 점 중에 하나가 라우팅 테이블 관리이다. DSR의 경로 캐시는 일정시간이 지나면 갱신하는 게 아니라 일정 수 이상의 경로가 저장되었을 때만 새로이 갱신이 된다. AODV에서의 라우팅 테이블의 경로 정보는 이전에 얻어진 경로 정보에 관해서만 유지하고, Expire time

을 정하여 라우팅에 대한 유효시간을 설정한다. hash 값은 각 노드에서 생성한 값을 저장함으로서, RREP의 패킷에 대한 인증 시 사용된다.

Ⅲ. 시뮬레이션

시뮬레이션은 상세한 프로토콜 모델링과 성능 분석을 목표로 가장 널리 사용되고 있는 NS-2를 채택하였다. 논문의 주제가 라우팅의 보안 이슈이기 때문에 결과를 제대로 보기 위하여 노드 수를 3개로 제한하였고, 확장도 가능하다. 인증에는 Secure Hash Algorithm(SHA1), HMAC-MD5, HMAC SHA-1을 참조 하였다.

기본 가정으로 각 노드가 Key 테이블을 가지고 있다고 가정하였다. 또 Delay는 목적지와 출발지에만 가지고 있다고 가정하였다. 출발지 노드는 0이고 목적지 노드는 2로 설정하였다.

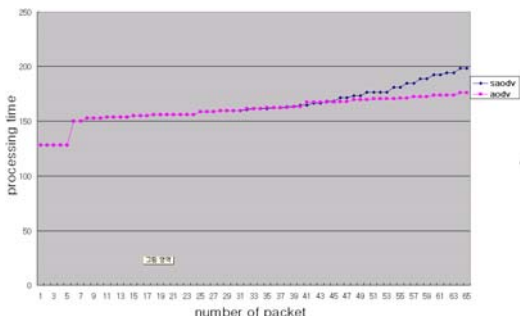

그림 10. AODV와 S-AODV의 처리시간 비교

이 그림은 패킷에 따른 진행시간을 AODV와 S-AODV로 비교하였다. 그림에서도 알 수 있듯이 X축이 41까지는 거의 비슷하나 갈수록 인증하는데 시간이 걸리기 때문에 늘어나는 것을 볼 수 있다. 위 결과에서도 볼 수 있듯이 실행 결과가 많이 차이는 나지 않는다.

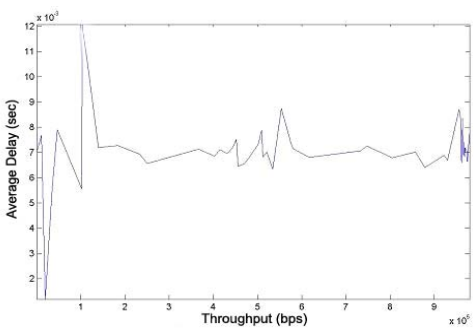

그림 11. AODV 평균 지연시간

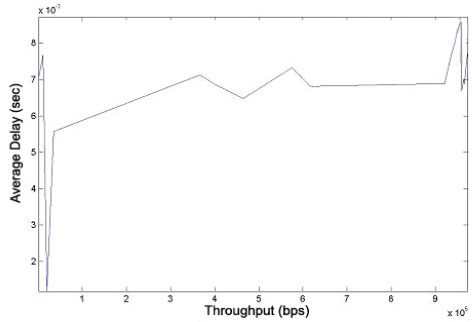

그림 12. S-AODV 평균 지연시간

위에 결과에서도 볼 수 있듯이 AODV와 S-AODV 간의 평균 지연을 비교한 그림이다. 그림에서 보듯이 노드간의 평균 지연 차이는 크지 않다. 하지만, 약간의 지연차가 나는 이유는 패킷의 크기가 인증으로 커지기 때문에 한번 지연이 생기면 계속 누적되기 때문이다. 그렇기 때문에 AODV에서 보이는 편차보다 더 완만한 곡선을 그린다.

Hash chain, HMAC, 그리고 Keytable로서 인증에 대한 구현을 하였다. 하지만 보안은 시나리오를 통해서 증명하기가 쉽지 않기 때문에 실습을 통한 결과로 보안을 적용했을 때보다 얼마나 지연이 생겼는지 밖에 추출할 수밖에 없었다. 그리고 위의 인증 방법은 아직은 결과가 검증이 되질 않았다는 단점이 있고 키 배포와 delay를 목적지에게 전달해주는데 문제점이 있다.

Ⅳ. 결 론

본 논문에서는 이동 애드혹 망을 위해 시공간 방식을 기반으로 하는 보안 라우팅 프로토콜의 설계 및 성능을 기술하였다. 제안된 시공간 방식은 시간영역에서는 출발지와 목적지 간에 키 배포를 수행하고, 또한 공간영역에서는 경로 상에서의 침입 탐지를 수행한다. 데이터 인증을 위해 효율성이 높은 대칭키 방식을 사용하고, 비밀 키는 출발지에서 목적지로 시간차를 이용하여 배포되어 진다. 또한, 단방향의 해쉬체인을 공간측면에서 홉단위로 형성함으로써, 침입자나 포섭된 노드가 라우팅 정보를 조작하는 것을 방지한다. 제안된 보안 라우팅 프로토콜인 S-AODV의 성능을 평가하기 위해 ns-2 시뮬레이션을 통하여 동일 조건하에서 기존의 AODV와 비교한 결과, 우리가 제안한 프로토콜이 보안 기능을 지원하면서도 동시에 지연시간과 처리율 관점에서 기존의 AODV와 유사한 성능을 보여주고 있다.

키 테이블은 항상 모든 노드들이 가지고 있어야 하는데 키 관리자가 있거나 혹은 다른 루트를 통해서 보내져야만 한다. 실행은 안전한 채널이라는 가정하에 보냈지만 실제 상황에서는 좀 더 보완해야 하고, 키 테이블을 입수한 공격자가 모두 대입하여 값을 얻지 못하도록 크기를 늘리거나 다른 방법이 필요하다. 그리고 delay time을 전달해주는데 좀 더 효율적인 전달 방법을 연구해야 한다. 이러한 문제와 향후 다양한 인증 알고리즘을 통한 비교 분석 및 공격 시나리오를 적용하여 좀 더 낳은 검증 및 개선을 하도록 할 것이다.

참 고 문 헌

[1] 권혜연,신재욱,이병복,최지혁,남상우,임선배, "이동 Ad hoc 네트워크 기술 동향", 전자통신동향분석, 제18권 제2호, 2003.

[2] L. Zhou and Z.J. Haas. "Securing Ad Hoc Networks", IEEE Network Magazine, November 1999.

[3] M.G. Zapata and N. Asokan, "Securing ad hoc Routing Protocols", Wise'02, Atlanta, Georgia, USA, September 28, 2002.

[4] D.B. Johnson, D.A. Maltz, Y. Hu, and J.G. Jetcheva, "The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks", IETF Internet Draft, draft-ietf-manet-dsr-07.txt, Work in progress, February 2002.

[5] Y. Hu, D.B. Johnson, and A. Perrig, "SEAD: Secure Efficient Distance Vector Routing for Mobile Wireless Ad Hoc Networks", Proceedings of the 4th IEEE Workshop on Mobile Computing Systems & Applications (WMCSA 2002), pp. 3-13, IEEE Calicoon, NY, June 2002.

[6] C.E. Perkins, E.M. Belding-Royer, and S. Das. "Ad Hoc On Demand Distance Vector (AODV) Routing", IETF RFC 3561, July 2003.

[7] M.G. Zapata. "Secure Ad hoc On-Demand Distance Vector (SAODV) Routing", IETF Internet Draft, draft-guerrero-manet-saodv-00.txt, August 2001.

[8] P. Cheng and R. Glenn, "Test Cases for HMAC-MD5 and MAC-SHA-1", IETF RFC 2202, September 1997.

[9] D. Eastlake and P. Jones, "US Secure Hash Algorithm 1 (SHA1)", IETF RFC 3174, September 2001.

조 인 휘 (Inwhee Joe) 정회원

1983년 2월 한양대학교 전자 공학과 졸업

1994년 12월 미국 University of Arizona, Electrical and Computer Engineering, M.S.

1994년 12월 미국 Georgia Tech, Electrical and Computer Engineering, Ph.D.

1992년 12월 (주) 데이콤 종합연구소 선임연구원

2000년 6월 미국 Oak Ridge 국립연구소 연구원

2002년 8월 미국 Bellcore Lab (Telcordia) 연구원

2002년 9월~현재 한양대학교 정보통신학부 부교수

<관심분야> Mobile Internet, Cellular System and PCS, RFID/USN, Mobility Management