韓國컴퓨터情報學會 論文誌
第16卷 第3號, 2011. 3.

2011-16-3-5-2

멀티홉 클러스터 센서 네트워크 환경 기반에서 견고한 키 교환

한 승 진*

A Robust Pair-wise Key Agreement Scheme based on Multi-hop Clustering Sensor Network Environments

Seungjin Han*

요 약

본 논문에서는 신뢰하는 제 3기관(혹은 장치)(TTP)이 없고 사전에 키가 분배되지 않는 멀티 홉 클러스터 센서 네트워크 환경에서 안전하게 암호화된 키를 교환하는 알고리즘을 제안한다. 기존의 연구는 TTP가 존재하거나 노드 간 키가 이미 분배되었다는 가정하에서 진행되었다. 그러나, 기존의 방법들은 기반 구조가 없는 USN 환경에서는 가능하지 않다. 기존 연구 중 일부는 난수를 이용한 Diffie-Hellman 알고리즘을 이용하여 문제를 해결하고자 하였으나 재생 공격과 중간자 공격에 취약한 것으로 나타났다. 기존의 Diffie-Hellman 알고리즘에서 취약한 문제로 드러난 노드 간 인증 문제는 μTESLA를 사용한 수정된 Diffie-Hellman 알고리즘으로 해결한다. 본 논문에서는 수정된 Diffie-Hellman 알고리즘에 타임 스탬프를 사용한 일회용 패스워드(OTP)를 추가하여 안전하면서, 가볍고, 강인한 키 교환 알고리즘을 제안한다. 마지막으로 인증, 기밀성, 무결성, 부인방지, 후방향 안전성 및 전방향 안전성에 대해서 안전하다는 것을 검증한다.

▸ Keyword : 센서네트워크, 키교환, 중간자공격, 재생공격, 클러스터기반, 인증, 기밀성, 데이터무결성, 부인방지, 후방향안전성, 전방향안전성, 시간

Abstract

In this paper, we proposed a scheme that it safely exchanges encrypted keys without Trust Third Party (TTP) and Pre-distributing keys in multi-hop clustering sensor networks. Existing research assume that it exists a TTP or already it was pre-distributed a encrypted key between nodes. However, existing methods are not sufficient for USN environment without infrastructure. Some existing studies using a random number Diffie-Hellman algorithm to solve the problem. but the method was vulnerable to Replay and Man-in-the-middle attack from the malicious nodes. Therefore, authentication problem between nodes is solved by adding a μTESLA. In this paper, we propose a modified Diffie-Hellman algorithm that it is safe, lightweight, and robust pair-wise agreement algorithm by

• 제1저자 : 한승진
• 투고일 : 2010. 11. 05, 심사일 : 2010. 12. 03, 게재확정일 : 2010. 12. 10.
* 경인여자대학 e-비즈니스과 부교수(Asscociate Prof., Dept. of e-Business, Kyungin Women's College)
※ 본 연구는 2009학년도 경인여자대학 교내연구지원 연구비에 의해 수행되었음

adding One Time Password (OTP) with timestamp. Lastly, authentication, confidentiality, integrity, non-impersonation, backward secrecy, and forward secrecy to verify that it is safe.

▸ Keyword : Sensor Networks, Diffie-Hellman, Pair-wise Key, Man-in-the-middle Attack, Replay Attack, OTP(One Time Password), Cluster-Based, Authentication, Confidentiality, Integrity, Non-impersonation, Backward secrecy, Forward secrecy, Timestamp

Ⅰ. 서 론

유비쿼터스 시대의 기반 기술 중에 하나인 센서 네트워크는 고정된 기반 시설(Infrastructure) 없이 이동 노드간 경로를 설정하여 신호를 주고 받는 무선 네트워크를 의미한다. 고정된 기반 시설이 없다는 것은 일반적인 무선 네트워크(Infrastructured Wireless Networks)와는 또 다른 성격의 네트워크를 의미한다. 여기서 노드란 RFID(Radio Frequency IDentifier) 칩을 이용하여 다른 고정된 장치 혹은 시설물에 부착하여 신호를 전송하는 장치를 의미할 수 있고, 또한 노트북, 스마트폰, 자동차, 센서 칩이 부착된 물건 혹은 기타 이동 장치에 부착하여 해당 노드가 이동하면서 장치간 신호를 주고 받을 수도 있다.

고정된 기반 시설이 없는 무선 네트워크에서의 이동 노드간 신호의 송수신은 기존의 기반 시설이 있는 무선 네트워크에 비해서 보안상 많은 취약점을 내포한다. 이러한 사실은 네트워크 내부에 악의의 목적을 가진 노드가 존재한다면 네트워크 전체의 성능에 커다란 영향을 미칠 수 있다.

[1,2]에서는 소스 노드로부터 해당 클러스터의 헤더까지 1 홉으로 이루어져 있다. 그러나 본 논문에서는 소스 노드와 클러스터 헤더까지 멀티 홉으로 이루어진 클러스터 기반의 센서 네트워크이다. 따라서, 1 홉 클러스터 기반의 센서 네트워크에서보다 멀티 홉 클러스터 기반의 센서 네트워크의 보안 문제가 더욱 어렵다[1,3-5].

센서 네트워크에서 임의의 노드가 악의의 의도를 갖게 되는 경우 이 노드로부터 패킷을 보호할 수 있는 장치 혹은 방법이 필요하다. 이에 대한 해결 방법의 하나로 노드간 주고받는 패킷을 암호화하는 방법이 있다. 유선망에서는 두 노드가 신뢰하는 Trusted Third Party (TTP)를 통해 인증을 받고, 대칭키 혹은 공개키를 전달 받는다. 기존의 대부분의 방법[3-5]은 키를 이미 안전하게 분배했다고 가정한다. 그러나 센서 네트워크 환경에서는 신뢰할 만한 TTP를 둔다는 것이 센서 네트워크 구조상 불가능하다.

기존의 논문[6]에서는 클러스터 헤더 노드를 TTP로서 역할을 하도록 하는 방법도 있으나, 이는 해당 노드의 전력 문제와 클러스터 헤더의 성능 상의 문제를 야기할 수 있고, 일반 노드들이 클러스터 헤더로서 역할을 거부하는 문제점이 발생할 수 있다. 또한 모든 노드들에게 사전에 대칭 키 혹은 공개 키를 분배한다는 것도 쉬운 일은 아니다.

본 논문에서는 [1,7,12]에서 제안한 방법을 개선하여 Diffie-Hellman의 키 교환 알고리즘[8]을 이용하여 별도의 TTP를 두지 않으면서, 사전에 노드들에게 키를 배포하지 않고도 노드 간에 안전하게 키를 주고 받는 방법이 멀티 홉 클러스터 기반의 센서 네트워크 환경에서 가능하도록 제안한다. 또한 Diffie-Hellman을 이용한 키 교환 알고리즘에서 대표적으로 알려진 취약점을 보완하기 위해 μTESLA와 암호화된 시간을 이용하는 One Time Password (OTP)를 이용한다. 또한 시간을 이용한 OTP를 통해 악의의 노드를 검출하고 경로 참여에 배제하는 알고리즘을 제안한다. 이와 같은 방법은 센서 네트워크에서 뿐만 아니라 유비쿼터스 센서 네트워킹(USN)에서 센서 노드들도 사용이 가능한 암호화 방법으로써 노드들 간의 안전한 패킷 송수신이 필요한 모든 분야에서 적용이 가능하다.

Ⅱ. Diffie-Hellman 알고리즘의 센서 네트워크 적용

센서 네트워크는 기존의 무선 네트워크에 비해 악의적인 공격에 매우 취약하다. 센서 네트워크에서 위협과 공격은 공격자의 능력, 공격자의 접근 수준, 그리고 공격자의 간섭 수준에 기반을 둔 몇 가지로 분류된다[9].

공격자는 센서 노드와 동일한 네트워크내에서 동일한 모델을 사용하여 센서 노드의 에너지와 처리 능력과 같은 자원을 소모하도록 한다.

공격자는 마이크로 컴퓨터, 노트북, 혹은 전파 장치가 탑재된 장치를 이용하여 특정 센서 노드의 전파 세기보다 강하게 작동하여 에너지 공급, 처리 능력, 그리고 메모리의 소모를 높이고 센서 노드간의 통신 속도를 지나치게 느리게 한다.

공격은 내부 혹은 외부로부터 공격이 있다. 외부 공격은

네트워크상에서 도청과 같은 형태로 이루어지기 때문에 탐지하기 매우 어렵지만, 센서 노드간의 강력한 암호를 통해 이러한 공격으로부터 방어가 가능하다. 외부 공격은 네트워크 패킷을 가로채거나 수정할 수도 있고, 고의로 잘못된 패킷을 네트워크로 흘려보낼 수도 있다. 내부 공격은 네트워크의 적법한 사용자에게 악성 코드를 전송하는 경우이다.

센서 네트워크에서 공격은 비 침투 공격(non-invasive)과 침투 공격(invasive)으로 나눌 수 있다. 비 침투 공격은 전력, 시간, 혹은 주파수와 같은 부채널 공격으로 구성되고, 침투 공격은 DoS(Denial of Service), 전송 중 정보공격, 노드 응답 공격, 경로 공격 등으로 나눌 수 있다.

2 장에서는 우선 본 논문의 환경과 사용 기호에 대해서 설명하고, 기존의 Diffie-Hellman 알고리즘을 센서 네트워크에 적용 시 발생할 수 있는 문제점에 대해서 설명한다.

2.1 환경

그림 1은 본 논문에서 제안하는 알고리즘을 적용하기 위한 멀티 홉 클러스터 기반의 센서 네트워크이고, 클러스터 1에 위치한 소스 노드(A)에서 클러스터 2에 위치한 목적 노드(B)로 패킷을 전송하는 경우이다.

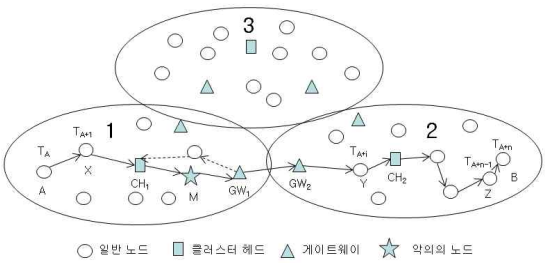

그림 1. 본 논문의 네트워크 구조
Fig 1. Network Environments

노드들 중에 한 노드를 선출하여 클러스터 헤드로 정한다. 여기서 클러스터 헤드 선출은 기존의 방법(LEACH[10], LEACH-C[11], HEED[12], 2DE[13]) 중 하나를 이용한다. 노드는 다른 노드와 패킷 송수신 시 항상 클러스터 헤드를 통해서 송수신한다. 일반 노드들과 클러스터 헤드간의 거리는 1홉이다. 또한 노드가 특정 클러스터에 진입하는 경우 클러스터 헤드에게 인증을 받아야 한다. 클러스터 헤드간의 인증은 클러스터 헤드와 노드간의 인증과 마찬가지로 시간을 이용한 OTP가 적용된 Diffie-Hellman 알고리즘이 적용된 방법을 이용하여 상호 인증한다.

2.2 용어정리

다음은 본 논문의 개념을 설명하는데 필요한 용어들의 정리[1]이다.

A, B : 노드
M : 악의의(Malicious) 노드
g : 곱셈 군(multiplicative group) Z_p^*의 생성자(generator), p의 원시근(primitive root)
p, q : 강한 소수(strong prime), p = 2 × q + 1
a : 노드 A의 개인 키(private Key)
$g_a \leftarrow g^a \bmod p$: 노드 A의 공개 키(Public Key)
S_n : n 번째 세션 키
S'_{AB} : 노드 A와 노드 B가 이전 세션에서 공유하여 사용하던 키
$H(\ )$: 단방향 해쉬 함수
K_{AB} : 노드 A와 B가 공유하는 비밀키(K)
$y \leftarrow K_{AB}\{x\}$: A가 비밀키K 를 이용하여 x 를 y 로 암호화 혹은 복호화하여 B에게 전송
N_B : 노드 B에서 전송한 난수
$T_{N_{AB}}$: 노드 A와 B의 이전 세션 단계에서 종료된 시간
T_{thld} : 각 노드가 수용하는 시간 임계치

2.3 Diffie-Hellman 알고리즘의 문제점

Diffie-Hellman 방법은 기존에 많은 문헌[1,3-5]에서 설명이 되어있기 때문에 설명은 생략하고, 현재 많은 연구를 통해서 밝혀진 대표적인 문제점에 대해서 알아본다.

2.3.1 취약점

센서 네트워크는 노드들간 무선을 이용하기 때문에 보안에 매우 취약하다. 이를 극복하기 위해 노드간 비밀 키를 사용한다. 비밀 키 전달 방법의 하나로 Diffie-Hellman 방법을 적용한 선행 연구가 많이 있다. 그러나 Diffie-Hellman 방법은 중간자 공격과 재생 공격에 취약하다. 즉, 클러스터 내부 혹은 클러스터간에 노드끼리 송수신하는 패킷을 중간의 노드가 가로채서 이를 이용하여 목적지 노드에게는 소스 노드로, 소스 노드에게 는 목적지 노드로 가장하는 중간자 공격과 노드간의 세션을 중간에 가로채어 이를 이용하여 상대방에게 접속을 시도하는 재생 공격이 가능하다.

2.3.2 중간자 공격(Man-in-the-Middle Attack)

그림 2는 그림 1에서 발생 가능한 중간자 공격을 나타낸 것이다.

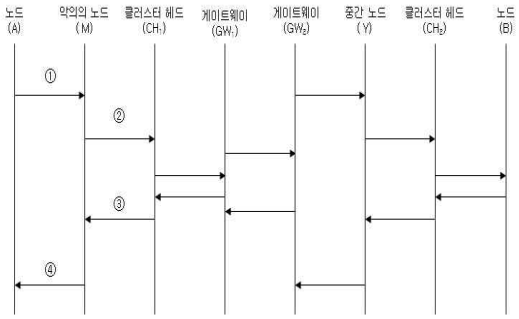

그림 2. 클러스에서 중간자 공격
Fig 2. Man-in-the-middle Attack in cluster

① 노드 A는 $a \in_A [1, p-1]$을 생성하고, $g_a \leftarrow g^a \pmod p$를 계산한다. 노드 A는 g_a를 중간 노드를 거쳐 노드 M에게 전송한다.

② 노드 M은 $m \in_M [1, p-1]$를 선택해서 $g_m \leftarrow g^m \pmod p$를 계산한다. 노드 M은 g_m을 자신이 속한 클러스터 헤드 CH_1에게 전송한다. 노드 M에게서 g_m을 수신한 클러스터 헤드 CH_1은 게이트웨이 GW_1을 통해 클러스터 2에 있는 노드 B에 전송한다.

③ 클러스터 2에 있는 노드 B는 $b \in_B [1, p-1]$을 생성하고, $g_b \leftarrow g^b \pmod p$를 계산한다. 노드 B는 g_b를 클러스터 헤드 CH_2에게 전송한다. 노드 B에게서 g_b를 수신한 클러스터 헤드 CH_2는 게이트웨이 GW_2를 통해 클러스터 1에 있는 노드 M(노드 A)에게 전송한다.

④ 노드 M은 노드 A에게 ②와 같은 방법을 이용하여 g_m을 생성한 후 전송한다.

* 노드 A는 $S_1 \leftarrow g_m^a \pmod p$를 계산한다.

(이때 S_1은 노드 M이 $S_1 \leftarrow g_a^m \pmod p$를 계산할 수 있기 때문에 노드 A와 노드 M이 공유하는 것이 가능하다.)

* 노드 B는 $S_2 \leftarrow g_m^b \pmod p$를 계산한다.

(이때 S_2는 노드 M이 $S_2 \leftarrow g_m^b \pmod p$를 계산할 수 있기 때문에 노드 M과 노드 B가 공유하는 것이 가능하다.)

①~④의 단계를 통해 악의의 노드 M은 S_1과 S_2를 계산할 수 있기 때문에 노드 A에게는 노드 B처럼 위장이 가능하고, 노드 B에게는 노드 A처럼 위장이 가능하다. 따라서, 노드 M은 위의 방법을 통해 노드 A와 B에게 중간자 공격이 가능하다.

2.3.3 재생 공격(Replay Attack)

그림 3은 그림 1에서 발생 가능한 재생 공격을 나타낸 것이다.

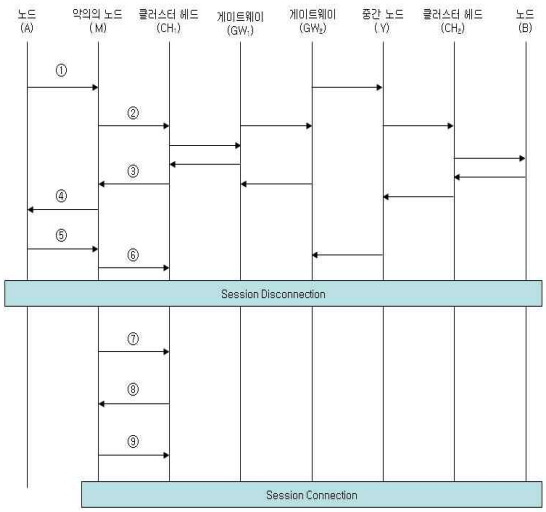

그림 3. 클러스에서 재생 공격
Fig 3. Replay Attack in cluster

① $\{A \rightarrow M(B) : S'_{AB}\}_{K_{AB}}$

노드 A는 노드 M(B)에게 A와 B가 공유하는 비밀키 K_{AB}를 이용하여 세션키 S'_{AB}를 전송한다.

② $\{M(A) \rightarrow B : S'_{AB}\}_{K_{AB}}$

노드 M(A)은 노드 B에게 A가 B와 공유하는 비밀키 K_{AB}를 이용하여 암호화한 세션키 S'_{AB}를 전달한다.

③ $\{B \rightarrow M(A) : N_B\}_{S'_{AB}}$

노드 B는 세션 키 S'_{AB}를 이용하여 난수 N_B를 암호화하여 노드 M(A)에게 전송한다.

④ $\{M(B) \rightarrow A : N_B\}_{S'_{AB}}$

노드 M(B)은 세션 키 S'_{AB}를 이용하여 암호화된 난수 N_B를 노드 A에게 전달한다.

⑤ $\{A \rightarrow M(B) : N_B - 1\}_{S'_{AB}}$

노드 A는 노드 M(B)에게 세션 키 S'_{AB}를 이용하여 난수 $N_B - 1$를 암호화하여 전송한다.

⑥ $\{M(A) \rightarrow B : N_B - 1\}_{S'_{AB}}$

노드 M(A)은 노드 B에게 세션 키 S'_{AB}를 이용하여 암호화된 난수 $N_B - 1$를 전달한다. 세션이 종료된다.

⑦ $\{M(A) \rightarrow B : S'_{AB}\}_{K_{AB}}$

노드 M은 노드 B에게 이전에 노드 A와 B와의 연결에서 이용되었던 비밀키 K_{AB}와 노드 A와 노드 B와의 이전 세션에서 사용되었던 세션키 S'_{AB}를 전송한다.

⑧ $\{B \rightarrow M(A) : N_B\}_{S'_{AB}}$

노드 B는 세션 키 S'_{AB}를 이용하여 난수 N_B를 암호화하여 노드 M(A)에게 전송한다.

⑨ $\{M(A) \rightarrow B : N_B - 1\}_{S'_{AB}}$

노드 M(A)은 노드 B에게 이전 연결에서 사용되었던 세션 키 S'_{AB}를 이용하여 암호화된 난수 $N_B - 1$를 전달한다. 노드 B는 노드 M을 노드 A로 오인하여 세션이 연결된다.

⑦ ~ ⑨의 단계를 통해 악의의 노드 M은 자신이 보관하고 있는 K_{AB}와 S'_{AB}를 이용하여 필요에 따라 노드 A 혹은 B와 세션 연결이 가능하다. 이러한 방법이 가능한 이유는 난수를 이용하여 OTP를 만들었기 때문에 노드 M은 ⑦ ~ ⑨의 단계를 이용해 노드 A와 B에게 이전에 사용했던 난수를 이용한 세션 키를 이용하여 재생 공격이 가능하다.

따라서, 본 논문에서는 Diffie-Hellman 알고리즘을 멀티홉 클러스터 기반의 센서 네트워크에서 키 교환 알고리즘으로 적용하지만, 위의 두 가지 문제점을 해결하기 위해 노드와 클러스터 헤드간 또는 클러스터 헤드간의 인증시 μTESLA[14] 방식을 이용하여 상호 인증을 하도록 하고, 시간이 적용된 OTP를 이용하여 두 가지 공격에 대해서 강인한 키 교환 알고리즘을 제안한다.

본 논문의 타당성을 검증하기 위해 [4]에서 검증한 방법을 이용하여 인증(Authentication), 기밀성(Confidentiality), 데이터 무결성(Integrity), 부인방지(Non-impersonation), 후방향 안전성(Backward secrecy), 전방향 안전성(Forward secrecy)에 대해서 검증한다.

III. 멀티 홉 클러스터 기반의 센서 네트워크에서 안전한 키 교환

본 논문의 모든 노드는 자신의 인증 검사에 대한 무결성을 위해 μTESLA를 이용한다. μTESLA는 노드들간의 시간적인 동기화가 필요하다. 유선망에서는 TTP를 통해 시간 동기화가 이루어지지만, 본 논문의 환경에서는 TTP가 존재하지 않으므로 μTESLA에서 필요한 노드들 간의 시간 동기화는 일반 노드와 클러스터 헤드간 주고 받는 메시지내의 시간을 이용하여 동기화 한다. 본 논문에서는 센서 노드에서 적용되는 μTESLA처럼 모든 노드들 간의 시간 동기화가 필요 없고, 오직 일반 노드와 클러스터 헤드간 만이 동기화가 필요하다. 본 논문에서는 기존의 Diffie-Hellman 방법을 수정하여 클러스터 기반의 센서 네트워크에서 강인한 키 교환 시스템을 제안한다.

임의 노드 A는 특정 클러스터에 조인 시 해당 클러스터로부터 클러스터 헤드(CH_A)의 공개 키(g_{CH})를 비콘(Beacon) 메시지를 통해 전송받는다. 노드 A는 자신의 정보를 클러스터 헤드(CH_A)의 공개 키(g_{CH})를 이용하여 클러스터 헤드에게 전송하고 클러스터 헤드는 인증을 한다.

노드 A는 개인키 $a \in_A [1, p-1]$를 선택하고, B의 공개 키 g_b를 이용하여 노드 A와 B 사이에 사용할 비밀 키와 노드 A의 ID를 암호화 한다. 시간 TA를 단방향 해쉬 함수에 적용시키고, 노드 A의 개인 키(a)로 암호화하여 시간 TA를 추가하여 (1)과 같은 메시지를 노드 X로 송신한다.

$$\{A \rightarrow X : g_b(ID_A, K_{AB}), K_{AB}(a(H(T_A)), T_A), Null, Null\}$$
$$where, \Phi_0 = (g_b(ID_A, K_{AB}), K_{AB}(a(H(T_A))), T_A), Null, Null) \quad \cdots\cdots (1)$$

그림 1처럼 수식 (1)을 수신한 노드 X는 자신의 개인 키로 암호화한 시간 T_{A+1}을 추가하여 다음 노드인 클러스터 1의 헤드인 CH_1로 메시지를 전달한다.

$$\{X \rightarrow CH_1 : \Phi_0, g_b(ID_X, T_{A+1}), x(T_{A+1})\} \quad \cdots\cdots (2)$$
$$where, \Phi_1 = (\Phi_0, g_b(ID_X, T_{A+1}), x(T_{A+1}))$$

$$\{CH_1 \rightarrow M : \Phi_1, g_b(ID_{CH_1}, T_{A+2}), ch_1(T_{A+2})\} \quad \cdots\cdots (3)$$
$$where, \Phi_2 = (\Phi_1, g_b(ID_{CH_1}, T_{A+2}), ch_1(T_{A+2}))$$

$$\{M \rightarrow GW_1 : \Phi_2, g_b(ID_M, T_{A+3}), m(T_{A+3})\} \quad \cdots\cdots (4)$$
$$where, \Phi_3 = (\Phi_2, g_b(ID_M, T_{A+3}), m(T_{A+3}))$$

$$\{Y \rightarrow CH_2 : \Phi_{i-1}, g_b(ID_Y, T_{A+i}), y(T_{A+i})\} \quad \cdots\cdots (5)$$
$$where, \Phi_i = (\Phi_{i-1}, g_b(ID_Y, T_{A+i}), y(T_{A+i}))$$

$$\{Z \rightarrow B : \Phi_{i+n-2}, g_b(ID_Z, T_{A+n-1}), z(T_{A+n-1})\} \quad \cdots\cdots (6)$$

(수식 3)에서 악의의 노드 M은 CH_1에게서 전송 받은 메시지에서 g_{CH_1}을 이용하여 T_{A+2}를 수정하고, 자신이 GW_1에게 전송하는 $g_b(T_{A+3}), m(T_{A+3})$에서 잘못된 시간을 입력하여 다음 노드인 GW_1에게 전송한다. (수식 4)를 전송받은 GW_1은 M의 공개 키 g_m를 이용하여 $m(T_{A+3})$를 복호화 하여 얻은 T_{A+3}과 자신의 시간(T_{A+4})와 비교를 한다. $T_{A+3} < T_{A+4}$라면 (수식 4)에 자신의 ID와 시간을 B의 공개키로 암호화하고, 또 시간을 자신의 개인 키로 암호화하여 다음 노드로 전송한다. $T_{A+3} > T_{A+4}$라면 즉, (수식 8)을 만족하지 않는다면, GW_1의 이전 노드가 잘못된 시간 정보를 전송하였기 때문에 GW_1은 자신이 속한 클러스터 헤드(CH_1)에게 CH_1의 공개 키를 이용하여 악의의 노드에 대한 정보를 (수식 7)과 같은 형식으로 전송한다.

$$\{GW_1 \rightarrow CH_1 : g_{CH_1}(ID_{GW_1}, ID_M, T_{A+4}), gw_1(T_{A+4})\}$$

..... (7)

$T_{A+3} < T_{A+4}$의 조건에서 노드 M에 의해 T_{A+3}이 T_{A+2}에 비해 상당히 큰 값을 입력했다면 이후 GW_1 이후의 노드들은 (수식 8)을 만족하지 않는 값이거나 혹은 노드 X가 전송한 시간에 비해서 현저히 큰 값이 노드 B에게 전송된다. 목적지 노드 B는 $b(g_b(K_{AB}))$를 통해 K_{AB}를 계산하고, K_{AB}를 이용하여 $a(H(T_A))$와 TA를 얻는다. A의 공개 키인 g_a를 이용하여 $H(T_A)$ 값을 계산하고 자신의 해쉬함수를 이용하여 $H(T_A)$를 계산하여 A로부터 받은 $H(T_A)$와 비교한다. 이후 노드 B는 자신의 개인 키와 중간 노드들의 공개 키를 이용하여 각 노드들이 전송한 시간을 검사한다. 검사 규칙은 (수식 8)과 (수식 9)를 만족해야 한다.

$$T_A < T_{A+1} < \ldots < T_{A+i} < \ldots < T_{A+n-1} < T_{A+n}$$

..... (8)

소스 노드 A와 목적지 노드 B가 이전에 서로 메시지를 송수신하여 <표 1>에 존재한다면 소스 노드 A의 시간 T_A는 $T_{N_{AB}}$보다 커야 한다. 즉, (수식 9)와 같은 조건을 만족해야 한다.

$$T_A > T_{N_{AB}} \quad \ldots\ldots (9)$$

여기서, $T_{N_{AB}}$는 소스 노드 A와 목적지 노드 B의 세션이 이전 단계에서 종료된 시간이다.

Algorithm 1. ***Generate secret key and send by source node***

Input : Destination node's Public key g, Source node's Timestamp T,
Source node's ID, Secret key K

Output : Encryption message with timestamp Φ

1: **procedure** ***Key_source*** $(g_{dest}, T_{src}, K_{SrcDest}, ID_{src})$
2: $\Phi \leftarrow \varnothing$
3: **Compute** $a_{src} \in_{SrcNode} [1, p-1]$
4 : $\Phi = (g_{dest}(ID_{src}, K_{SrcDest}), K_{SrcDest}(a_{src}(H(T_{src})), T_{src})$
5: ***SendtoNextNode***(Φ)
6: **if** (***ReceivingErrMsg***() $\notin \varnothing$) **and** (CHofSender $\neq$ CHofSourceNode) **then**
7: ***SendtoCH*** $(g_{Im's CH}(ID_{im}, ID_{im-1}, T_{im}), a_{im}(T_{im}))$
8: **end if**
9: **end procedure**

그림 4. 소스 노드에서의 키 생성 및 전송 알고리즘
Fig 4. Key generation and sending algorithm in source node

Algorithm 2. ***Check and generate message by immediate node***

Input : Received message Φ from before node, Immediate node's ID,
Immediate node's Timestamp T, Destination node's Public key g,
Before node's Public key g

Output : Encryption message with timestamp Φ or Error Message

1: **procedure** ***Key_immediate*** $(\Phi, g_{dest}, T_{im}, ID_{im}, g_{BeforeNode})$
2: **if** ***chooseID***(Φ) $\neq$ me **then** //In case of Immediate node
3: **if** ***right***$(\Phi,1) \in \varnothing$ **then**
4: **skip** // because before node is source node
5: **endif**
6: **else**
7: temp $\leftarrow$ ***right***$(\Phi,1)$
8: **if** $g_{BeforeNode}(temp) < T_{im}$ **then**
9: **Compute** $a_{im} \in_{ImNode} [1, p-1]$
10: **Add** $g_{dest}(ID_{im}, T_{im})$ **and** $a_{im}(T_{im})$ **to** Φ
11: **end if**
12: **else**
13: ***SendtoCH*** $(g_{Im's CH}(ID_{im}, ID_{im-1}, T_{im}), a_{im}(T_{im}))$
14: ***SendtoSourceNode*** $(g_{SrcNode}(ID_{im}, ID_{im-1}, T_{im}), a_{im}(T_{im}))$
15: **end else**
16: **end else**
17: **end if**
18: **else**
19: **call** ***Key_destination***() // In case of destination node
20: **end else**
21: **end procedure**

그림 5. 중간 노드에서 메시지 검사 및 생성 알고리즘
Fig 5. Message checking and generation algorithm in immediate node

Algorithm 3. ***Destination node receives the message***

Input : Received message Φ from before node, Immediate node's ID,
Immediate node's Timestamp T, Destination node's Public key g,

Before node's Public key g
Output : Encryption message with timestamp Φ or Error Message
1: **procedure** *Key_destination* ($\Phi, g_{BeforeNode}, T_{dest}$)
2: **if** *right*(Φ,1) $\in \varnothing$ **then**
3: **skip** // because before node is source node
4: **endif**
5: **else**
6: temp $\leftarrow$ *right*(Φ,1)
7: **if** $g_{BeforeNode}(temp) < T_{dest}$ **then**
8: **Get** $K_{SrcDest}$ **and** T_{src}
9: **if** $T_{src} > T_{dest}$
10: *SendtoCH* ($g_{Im'sCH}(ID_{im}, ID_{im-1}, T_{im}), a_{im}(T_{im})$)
11: *SendtoSourceNode* ($g_{SrcNode}(ID_{im}, ID_{im-1}, T_{im}), a_{im}(T_{im})$)
12: **end if**
13: **end if**
14: **else**
15: *SendtoCH* ($g_{Im'sCH}(ID_{im}, ID_{im-1}, T_{im}), a_{im}(T_{im})$)
16: *SendtoSourceNode* ($g_{SrcNode}(ID_{im}, ID_{im-1}, T_{im}), a_{im}(T_{im})$)
17: **end else**
18: **end else**
19: **end if**
20: **end procedure**

그림 6. 목적지 노드에서 메시지 검사 및 생성 알고리즘
Fig 6. Message checking and generation algorithm in destination node

[그림 4]부터 [그림 7]까지는 수식 (1)부터 (12)까지의 수식을 알고리즘으로 표현한 것으로써, 각 노드들이 메시지를 생성하고 에러 메시지를 처리하는 알고리즘이다.

특정 노드의 시간이 특정 노드 다음 노드의 시간보다 작거나, 또는 이전 노드들의 시간에 비해서 시간의 값이 현저히 크다면 목적지 노드는 해당 노드를 악의의 노드로 규정하고 (수식 10)과 같은 메시지를 통해 해당 노드 ID를 자신이 속한 클러스터 헤드에 보고하고, 클러스터 헤드(CH_2)는 자신이 속한 클러스터 멤버들에게 방송 메시지를 이용하여 악의의 노드를 포함하여 의심스러운 노드들을 알린다.

Algorithm 4. ***Cluster Head receives the error message***
Input : Received message Φ from before node, Immediate node's ID,
Immediate node's Timestamp T, Destination node's Public key g,
Before node's Public key g
Output : Encryption message with timestamp Φ or Error Message
1: **procedure** *MalNodeMng* ($\Phi, g_{BeforeNode}, T_{dest}$)
2: **GetReportNode**(Φ)
3: **Add** ReportNode **to** MalNodeMngTable
4: **GetMalCandiNode**(Φ)
5: **Add** MaliciousCandidateNode **to** MalNodeMngTable
6: **if** *count*(MaliciousCandidateNode, MalNodeMngTable) > *Threshold* **then**
7: ***BroadcastToCluster***(MaliciousCandidateNode)
8: **end if**
9: **end procedure**

그림 7. 클러스터 헤드에서 에러 메시지 처리 알고리즘
Fig 7. Error message handling algorithm in cluseter head

목적지 노드가 속한 클러스터 헤드(CH_2)는 소스 노드가 속한 클러스터의 헤드(CH_1)에게 보고 한다. 즉, (수식 11)과 같은 형식으로 악의의 노드가 속한 클러스터 헤드(CH_1)에 보고한다. (수식 11)은 노드 B가 A에게 악의의 노드를 포함하여 시간이 의심스러운 노드들을 보고한다. (수식 12)는 노드 A가 자신이 속한 클러스터의 헤드에 노드 B에게서 수신한 의심스러운 노드들을 보고하고, 이후에 클러스터 헤드(CH_1)는 자신이 속한 클러스터 멤버들에게 방송 메시지를 이용하여 악의의 노드를 포함하여 의심스러운 노드들을 알린다. 표 2는 클러스터 헤드가 관리하는 테이블이다. 클러스터 헤드는 악의의 노드 신고 횟수에 따라 주변 노드에게 주의 등급을 나누어 전송할 수 있다.

$$\left\{B \rightarrow CH_2 : g_{CH_2}(ID_B, ID_M, ID_{GW_1}, T_{A+n}), b(T_{A+n})\right\} \quad (10)$$

$$\left\{B \rightarrow A : g_A(ID_B, ID_M, ID_{GW_1}, T_{A+n}), b(T_{A+n})\right\} \quad \cdots\cdots (11)$$

$$\left\{A \rightarrow CH_1 : g_{CH_1}(ID_A, ID_M, ID_{GW_1}, T_{A+n+j}), a(T_{A+n+j})\right\}$$

$$\cdots\cdots (12)$$

소스 노드 A와 목적지 노드 B와의 세션 종료시간이 표 1에 존재하지 않는다면 목적지 노드 B는 소스 노드 A가 최초 전송

으로 간주한다. 그러나 $T_{N_{AB}}$는 (수식 13)을 만족해야 한다.

$$T_{thld} < T_{N_{AB}} \quad \cdots\cdots (13)$$

여기서, T_{thld}는 각 노드가 정한 다른 노드들과의 세션 종료 시간에 대한 임계치이다. 공격자는 키의 freshness를 위해 가급적 최근에 종료된 세션 키를 이용하려 할 것이다. 따라서 $T_{N_{AB}}$가 임계치 T_{thld} 범위를 벗어난다면, 목적지 노드는 재생 공격으로 간주하고 해당 메시지를 폐기한다.

표 1. 연결이 종료된 노드들과의 세션 종료 시간
Table 1. Session ended time with disconnected neighbor nodes

노드	세션 종료 시간
A	-
P	T_{thld}
Q	T_{thld}
X	$T_{N_{AX}}$
CH_1	$T_{N_{ACH_1}}$
M	$T_{N_{AM}}$
:	:
Y	$T_{N_{AY}}$
Z	$T_{N_{AZ}}$
B	$T_{N_{AB}}$

노드 A가 다른 노드와 세션이 종료된 후 <표 1>과 같은 테이블을 생성하고 관리한다.

표 2. 악의의 노드 관리(CH_1)
Table 2. Malicious node management in CH_1

노드	악의의 노드 보고 횟수	악의의 노드 보고 노드
A	0	
GW_1	0	
X	1	M
M	3	X, P, Q
P	1	M
Q	1	P
:	:	

여기서, $T_{N_{AX}}$, ... , $T_{N_{AB}}$는 노드 A와 각각 노드 Z와 B가 세션이 종료된 시간이다. 노드 A는 자신이기 때문에 세션 종료 시간이 없다. 노드 P와 Q는 A와의 세션 종료가 일정 시간이 지났기 때문에 T_{thld}로 초기화 된다.

기존 연구에서 살펴본 것처럼 Diffie-Hellman 방법은 중간자 공격과 재생 공격에 취약하다. 본 논문은 Diffie-Hellman을 이용하여 키 교환 시 중간자 공격을 위해서는 μTESLA 방식을 이용하여 상대 노드에게 자기 인증을 한다. 그러나 μTESLA 방식은 TESLA 방식에 비해 MANET 환경에는 적합하지만 노드들 간의 시간 동기화가 문제가 된다. 재생 공격을 위해서는 기존 연구[3]처럼 OTP를 이용하지만 난수를 이용한 OTP 방식은 중간의 악의의 노드에 의해 재생 공격이 가능하다[5].

따라서, 본 논문에서는 노드들의 암호화된 시간을 이용하여 노드들 자신의 인증에 사용하는 μTESLA의 시간 동기화 문제를 해결하고, 이러한 암호화된 시간은 재생 공격 방지를 위해 사용된다. 따라서 본 논문에서 제안하는 알고리즘을 이용하여 중간자 공격과 재생 공격으로부터 각 노드는 부인방지, 무결성과 기밀성을 보장받는다.

Ⅳ. 분석

본 논문에서 연구하는 시스템의 보안성을 평가하기 위해 다음과 같은 [4]에서 제시하는 요구사항 중 인증(Authentication), 기밀성(Confidentiality), 데이터 무결성(Integrity), 부인방지(Non-impersonation), 후방향 안전성(Backward secrecy), 전방향 안전성(Forward secrecy)에 대해서 검증한다.

5.1 인증(Authentication)

임의 노드 A는 특정 클러스터에 조인 시 해당 클러스터로부터 클러스터 헤드(CH_A)의 공개 키(g_{CH})를 비콘(Beacon) 메시지를 통해 전송받는다. 노드 A는 자신의 ID와 클러스터 헤드(CH_A)로부터 받은 공개키를 해쉬 함수에 적용하고, 자신의 개인 키로 암호화 한 식 (14)의 결과를 클러스터 헤드(CH_A)에게 전송한다.

$$Auc_{A,CH_A} = (ID_A, a(H(g_{CH}))) \quad \cdots\cdots (14)$$

식 (14)의 결과를 전송받은 클러스터 헤드(CH_A)는 ID를 통해 메시지가 노드 A로부터 전송 된 것을 안다. 클러스터 헤드(CH_A)는 노드 A의 공개키를 이용하여 $H(g_{CH})$를 얻고(식 (15)) 자신의 공개키를 해쉬 함수에 적용(식 (16))하여 비교한다.

$x = H(g_{CH})$ ..... (15)

$y = H(g'_{CH})$ ..... (16)

클러스터 헤드(CH_A)는 $x \equiv y$ 라면 노드 A를 적법한 노드로 인정한다.

5.2 기밀성(Confidentiality)

소스 노드 A가 목적지 노드 B에게 전송한 비밀 키 K_{AB}는 노드 A와 노드 B만이 공유하는 비밀 키로써, 비밀 키를 알지 못하는 다른 노드들은 K_{AB}로 암호화하는 내용을 복호화 할 수 없다.

5.3 데이터 무결성(Integrity)

각 노드 간에 주고 받는 메시지에는 시간이 적용된 OTP를 사용하기 때문에 설사 중간 세션 키가 노출되었다 하더라도 다음 세션에서는 비밀 키가 다시 변경되기 때문에 메시지를 변경할 수 없다.

5.4 부인방지(Non-impersonation)

각 노드들(클러스터 헤드 포함)은 자신의 시간을 자신의 개인 키로 암호화 한다. 따라서, 해당 노드의 개인 키가 노출되지 않는 한 해당 노드의 개인 키로 암호화하였기 때문에 해당 노드의 서명을 부인할 수 없다.

5.5 후방향 안전성(Backward secrecy)

데이터 무결성에서 분석한 것처럼 본 논문에서 제안하는 방법은 매번 메시지 송수신 시 시간이 포함된 OTP를 사용하기 때문에 세션 키가 노출되었다고 하더라도 노출된 세션 키를 이용하여 접속할 수 없다.

5.6 전방향 안전성(Forward secrecy)

후방향 안전성과 동일하다.

V. 결론 및 향후 연구과제

본 논문에서는 유비쿼터스 환경에서 핵심기술로 사용 중인 센서 네트워크에서 노드간에 TTP와 사전 키 분배 없이 시간을 이용하여 안전하게 비밀 키를 교환하는 방법에 대해서 제안하고, 분석을 통해 기존의 Diffie-Hellman 알고리즘의 문제점인 재생 공격과 중간자 공격에 대해 안전함을 입증하였다. 시간을 이용하여 악의의 목적을 갖는 노드를 신고하여 클러스터 내의 노드들이 악의의 노드를 배제하여 경로 설정이 가능하게 하였다. 또한 인증, 기밀성, 데이터 무결성, 부인방지, 후방향 및 전방향 안전성에 대해서 분석하여 안전함을 입증하였다.

본 논문의 결과는 센서 네트워크 환경에서 TTP가 없으면서 사전에 키 분배 없이 안전하고, 강인한 키 교환이 가능하고, 사용자 인증이나 세션 키 공유 및 확인을 요하는 시스템에 유용하게 적용이 가능하다. 또한 유비쿼터스 환경에서 자기 인증을 위해 필요한 μTESLA를 위해 시간 동기화 문제 해결의 한가지로 가능하다. 향후 클러스터 기반의 센서 네트워크에서 인증이 완료된 노드들 간에 클러스터 헤드를 거치지 않고 안전하게 통신할 수 있는 방안에 대해서 연구할 계획이다.

참고문헌

[1] Seungjin Han, J.H. Choi, “A Robust Pair-wise Key Agreement Scheme without Trusted Third Party and Pre-distributing Keys for MANET Environments,” Journal of The Korea Society of Computer and Information, KSCI, vol. 13, no. 5, Sep., 2008.

[2] J. S. Lee, et. al., "Energy Efficient Cluster Management Scheme for Ubiquitous Sensor Networks," International Conference on Computational Sciences and Its Applications, ICCSA 2008, pp. 73~83, 2008.

[3] S.H Seo, T.N Cho, and S.H Lee, “OTP-EKE: A Key Exchange Protocol based on One-Time-Password,” Journal of The Korean Institute of Information Scientists and Engineersm : System and Theory , vol. 29, no. 5, KIISE, June, 2002.

[4] C. C. Chang, K. C. Lin and J. S. Lee, "DH-Based Communication Method for Cluster-Based Ad Hoc Networks," 2nd International Conference on Mobile Technology, Applications and Systems, 15-17, Nov., 2005.

[5] Wenbo Mao, Modern Cryptography : Theory and Practice, Prentice Hall, July, 2003.

[6] S. Zhu, S. Xu, S. Setia, and S. Jajodia, "Establishment pair-wise keys for secure

communication in ad hoc networks: a probabilistic approach," In Proceedings of the 11th International Conference on Network Protocols, pp. 326~335, 2003.

[7] Seungjin Han, "A Pair-wise Key Agreement Scheme for Cluster-Based Sensor Networks," Journal of Kyungin Women's College, vol. 16, pp301-312, Kyungin Women's College, Jan., 2009.

[8] W. Diffie and M. Hellman, "New Directions on Cryptography," IEEE Transactions on Information Theory, IT-22(6): pp. 644~654, Nov., 1976.

[9] M. Healy, T. Newe, and E. Lewis, "Security for Wireless Sensor Networks: A Review," SAS 2009 - IEEE Sensors Applications Symposium, New Orleans, LA, USA, Feb., 17-19, 2009.

[10] W. R. Heinzelman, A. P. Chandrakasan, and H. Balakrishnan, "Energy-Efficient Communication Protocol for Wireless Microsensor Networks," Proc., 33rd Hawaii Int'l. Conf. Sys. Sci., Jan., 2000.

[11] W. R. Heinzelman, A. P. Chandrakasan, and H. Balakrishnan, "An Application-Specific Protocol Architecture for Wireless Microsensor Networks," IEEE Trans. Wireless Communication, vol. 1, no. 4, Oct., 2002.

[12] O. Younix and S. Fahmy, "Distributed Clustering in Ad-hoc Sensor Networks : A Hybrid, Energy-Efficient Approach," IEEE INFOCOM, Mar., 2004.

[13] J.S Kim, *Energy Efficient and Secure Cluster-based Routing Protocol in Wireless Sensor Networks*, Ph.D. Dissertation, Inha Univ., Feb., 2010.

[14] A. Hajami, K. Oudidi, and M. Elkoutbi, "A Distributed Key Management Scheme based on Multi hop Clustering Algorithm for MANETs," International Journal of Computer Science and Network Security, vol. 10, no. 2, pp.39~49, Feb., 2010.

저 자 소 개

한 승 진

1985~1990 : 인하대학교 이과대학 전자계산학과 학사

1990~1992 : 인하대학교 일반대학원 전자계산공학과 석사

1999~2002 : 인하대학교 전자계산공학과 박사

1992~1996 : 대우통신 종합연구소

1996~1996 : 한국전산원 초고속사업단

1996~1998 : SKTelecom 디지틀사업본부

2002~2004 : 인하대학교 컴퓨터공학부 강의조교수

2004~현재 : 경인여자대학 e-비즈니스과 부교수

2007~현재 : TTA PG103 표준화 위원

관심분야 : USN, MANET, Mobile Computing, 임베디드 시스템, Security, Middleware

E-mail : softman@kic.ac.kr