유비쿼터스 센서 네트워크에서 더블키를 이용한 경량 보안 프로토콜 설계 및 구현

정 연 일† · 이 승 룡††

요 약

유비쿼터스 컴퓨팅은 사용자에게 장소와 시간에 제약이 없이 자유롭게 네트워크에 접속 할 수 있는 환경을 제공하고 있다. 이러한 환경은 모든 정보의 공유 및 접근이 쉽게 이루어지는 반면, 인가되지 않은 사용자의 불법적인 접근도 쉽게 이루어질 수 있기 때문에 적합한 보안 정책이 필요하다. 특히 유비쿼터스 센서 네트워크의 센서 노드들은 제한된 전력을 이용하고 하드웨어적으로 작은 크기를 유지해야 하기 때문에 보안 정책 수립에 많은 제한이 발생하게 된다. 본 논문에서는 유비쿼터스 센서 네트워크에서 센서의 운영체제와 플랫폼, 라우팅 방식에 종속적이지 않은 더블키를 이용한 경량 보안 프로토콜을 제안한다. 본 논문에서는 더블키 방식을 제안하여 인증과 안전한 키 분배 및 교체가 이루어 질 수 있도록 한다. 보안 관리자가 네트워크 상황에 맞는 보안 레벨 변경 및 키 교체, 관리가 가능하기 때문에 적은 연산 처리만으로 최대의 보안 효과를 얻을 수 있는 장점이 있다. 성능 평가 결과 제안한 유비쿼터스 센서 네트워크에서 더블키를 이용한 경량 보안 프로토콜은 기존의 보안 정책 보다 상대적으로 저전력으로 보안 정책을 활용 할 수 있었다. 제안한 더블키를 이용한 경량 보안 프로토콜은 스마트 오피스 및 스마트 홈과 같은 실제 환경의 유비쿼터스 센서 네트워크에 적합하다고 할 수 있다.

키워드 : 유비쿼터스 센서 네트워크, 센서 보안, 센서 인증, 더블키 프로토콜

Design and Implementation of Double-Key based Light Weight Security Protocol in Ubiquitous Sensor Network

Yonil Zhung† · Sungyoung Lee††

ABSTRACT

Ubiquitous computing supports environment to freely connect to network without restrictions of place and time. This environment enables easy access and sharing of information, but because of easy unauthorized accesses, specified security policy is needed. Especially, ubiquitous sensor network devices use limited power and are small in size, so, many restrictions on policies are bound to happen. This paper proposes double-key based light weight security protocol, independent to specific sensor OS, platform and routing protocol in ubiquitous sensor network. The proposed protocol supports safe symmetric key distribution, and allows security manager to change and manage security levels and keys. This had a strong merit by which small process can make large security measures. In the performance evaluation, the proposed light weight security protocol using double-key in ubiquitous sensor network allows relatively efficient low power security policy. It will be efficient to ubiquitous sensor network, such as smart office and smart home.

Key Words : Ubiquitous Sensor Network(USN), Sensor Security, Sensor Authentication, Double-Key Protocol

1. 서 론

유비쿼터스 컴퓨팅은 사용자에게 장소와 시간에 제약이 없이 자유롭게 네트워크에 접속 할 수 있는 환경을 제공하고 있다. 이러한 환경은 모든 정보의 공유 및 접근이 쉽게 이루어지는 반면, 인가되지 않은 사용자의 불법적인 접근도 쉽게 이루어질 수 있기 때문에 적합한 보안 정책의 수립은 필수적이다. 유비쿼터스 컴퓨팅에서 무선 센서 네트워크는 가장 중요한 역할을 하는 분야 중에 하나이며 현재 많은 연구가 진행 중에 있다. 정보의 공유 및 접근이 쉽게 이루어지는 유비쿼터스 환경의 특징 때문에 유비쿼터스 센서 네트워크의 보안 분야는 다른 유비쿼터스 분야에 비하여 해결해야 할 문제점이 많이 발생하고 있다. 특히 유비쿼터스 센서 네트워크의 보안 분야는 무선을 이용하기 때문에 생기는 무선 보안 문제와 센서 노드의 제한된 환경에 맞는 새로운 보안 정책을 수립해야 하는 문제점을 동시에 가지고 있다. 본

※ 본 연구는 정보통신부 및 정보통신연구진흥원의 해외교수요원초빙사업의 연구결과로 수행되었음
† 준 회 원 : 경희대학교 컴퓨터공학과 박사과정
†† 종신회원 : 경희대학교 컴퓨터공학과 교수
논문접수 : 2007년 3월 30일, 심사완료 : 2007년 6월 12일

논문은 유비쿼터스 센서 네트워크의 두 가지 보안 문제 가운데 센서 노드의 제한된 환경으로 생기는 보안의 문제점에 대해서 논의하고 그 해결 방법을 제시하고자 한다.

유비쿼터스 센서 네트워크의 센서 노드들은 제한된 전력을 이용하고 하드웨어적으로 작은 크기를 유지해야 하기 때문에 연산 처리에도 제약이 따르게 된다. 그렇기 때문에 전력과 하드웨어의 제약을 고려하지 않은 기존의 보안 정책을 제약이 많은 유비쿼터스 센서 네트워크에 적용하기에는 문제점이 발생한다. 따라서 유비쿼터스 센서 노드들의 제한된 전력과 하드웨어를 고려한 인증 프로토콜, 경량 암호화 알고리즘 등의 새로운 보안 정책들이 필요하다. 이러한 제약 조건을 만족시키기 위해서 많은 보안 정책들이 연구 중에 있다. 하지만 이러한 보안 정책에는 각기 해결해야 하는 몇 가지 문제점들이 있다. 대칭키를 이용한 보안 정책을 사용 경우에는 암호화·복호화에 효율성을 보일 수 있지만 대칭키를 안전하게 공유해야 하는 문제점을 해결해야 한다. 브로드캐스트 메시지 인증의 경우 복잡한 절차가 요구되며 많은 트래픽을 발생시키기 때문에 많은 전력을 소모하게 된다. 대칭키를 이용한 보안 정책의 경우에는 특히 하나의 노드가 전복 되었을 경우에도 그 피해는 유비쿼터스 센서 네트워크 전체에 발생하게 된다. 공개키를 이용한 보안 정책을 사용할 경우에는 키 분배의 효율성을 보이지만 대칭키 보안 알고리즘에 비하여 보안 모듈의 크기가 크고 암호화와 복호화의 처리 시간이 길며 처리량이 많기 때문에 제한된 노드 환경에 적용하기 어렵다. 또한 키 생성 및 관리에 대한 모듈을 각 노드에서 포함하고 있어야 하며, 키 분배 및 인증 문제를 해결하기 위하여 키 관리 센터를 이용하는 연구도 진행 중이지만 이러한 방식 역시 새로운 트래픽을 발생시키며 노드 자체의 연산을 증가시키기 때문에 실제로 구현을 하여 활용하기에는 적당하다고 볼 수 없다. 센서 네트워크 보안에서는 대부분 상대적으로 모듈의 크기가 작고 암호화 처리 속도가 빠른 대칭키 보안 정책이 적용 되고 있다. 하지만 대칭키 기반의 보안 정책을 이용하기 위해서는 대칭키의 분배 문제가 가장 중요한 해결 사항이다. 또한 안전한 통신을 위해서는 주기적인 키 교체가 필수적인데 센서 노드의 제한된 하드웨어 환경에서는 구현하기가 쉽지가 않다. 대칭키의 안전한 분배와 주기적인 키 교체 문제를 해결하기 위하여 각각의 객체에서 같은 키를 생성하여 사용 하는 방법들이 연구되고 있지만 추가적인 모듈 포함과 연산은 센서의 제한적인 하드웨어에 부담이 되고 있다.

본 논문에서는 유비쿼터스 센서 네트워크의 제한된 센서 노드에 적합한 대칭키 기반의 더블키(Double-Key)를 이용한 경량 보안 프로토콜을 제안한다. 본 논문에서 새롭게 정의하여 사용하는 더블키 보안 정책은 대칭키 보안의 문제점인 안전한 키 갱신을 해결하기 위해서 제안 되었다. 더블키를 이용한 경량 보안 프로토콜은 더블키를 이용한 인증 프로토콜과 주기적 혹은 비주기적으로 대칭키의 교체가 가능한 더블키를 이용한 키 교체 관리 프로토콜로 이루어져 있다. 더블키 보안 프로토콜은 각 객체가 메인키(Main-key)와 보관키(Safety-Key)라고 명칭 하는 두 개의 대칭키를 소유하여 인증 및 암호화·복호화에 사용하며 새로운 대칭키를 갱신하는데 사용하는 정책이다. 본 논문에서 제안하는 더블키를 이용한 경량 보안 프로토콜은 다음과 같은 장점이 있다. 첫째, 각 개체 간 인증의 경우에 기존의 인증 방식을 이용하여 구현 할 경우 제한적인 하드웨어로 구성되어 있는 센서 노드들은 인증을 위한 알고리즘을 포함하고 있어야 한다. 이러한 방법의 접근은 센서 노드에서 연산량 증가를 가져오며 이러한 문제를 해결하기 위해서는 센서 노드 하드웨어적인 증가가 필요 하게 된다. 본 논문의 더블키를 이용한 보안 프로토콜은 경량화 된 인증 프로토콜을 제안하여 센서 노드에는 상대적으로 크기가 작은 모듈이 포함되기 때문에 하드웨어의 증가 없이 사용할 수 있다. 또한 센서 노드에는 인증을 위한 최소한의 연산량 증가로 안전한 인증을 할 수 있도록 하였다. 둘째, 기존의 공개키를 이용한 보안 정책과 대칭키를 이용한 보안 정책은 유비쿼터스 센서 네트워크에 적용하기에는 다음과 같은 문제점을 가지게 된다. 공개키 정책을 사용 할 경우 키 분배의 효율성이 있으나 키 관리의 어려움 및 연산량의 증가로 인한 전력 낭비가 심해지게 된다. 대칭키 정책을 사용할 경우 공개키 정책에 비하여 적은 연산량으로 처리가 가능하나 키 분배의 문제점을 해결해야 하며 하나의 키가 전복 될 경우 네트워크 전체가 전복되는 문제점이 발생하게 된다. 본 논문에서 제안하는 더블키를 이용한 경량 보안 프로토콜은 대칭키 기반의 보안 정책이기 때문에 상대적으로 적은 연산량으로 보안 처리가 가능하다. 또한 더블키를 이용한 키 분배 및 교체가 가능하며 센서 노드별로 다른 대칭키를 사용하여 대칭키 보안 정책의 단점인 키 분배 문제와 하나의 키가 전복되어 전체 네트워크에 영향을 주는 문제점을 해결하였다. 셋째, 같은 보안 효과를 가지고 있는 기존의 대칭키 기반 보안 프로토콜에 비하여 제안한 더블키를 이용한 경량 보안 프로토콜은 저전력의 장점을 보여 준다. 센서 노드들에 인증 및 보안 정책을 사용할 경우 인증을 위한 알고리즘, 키 관리 알고리즘과 같은 처리 코드가 포함이 되며 이러한 결과는 센서 노드에 많은 전력 소비를 가져올 수밖에 없다. 하지만 제안하는 프로토콜의 경우에는 대부분의 인증 및 키 생성의 연산 처리는 관리 서버에서 처리하고 실제 센서 노드에는 인증과 교체 알고리즘의 경량 모듈만 포함이 되기 때문에 기존의 보안 프로토콜에 비해서 저 전력으로 운영 할 수가 있다.

실제 성능 평가에서도 다른 센서 네트워크에서 사용하는 보안 정책보다 더 적은 보안 패킷의 증가율(7.9%)만으로도 보안 프로토콜을 구현 할 수 있었다. 안전한 인증과 키 교체 프로토콜에 소비되는 추가 전력이 적고(473.4 mA/year) 또한 센서 노드의 키 생성, 관리, 교체의 알고리즘을 제외함으로써 센서 노드의 연산량을 줄여서 저전력의 효과를 가져왔다. 이러한 결과는 제안하는 더블키를 이용한 경량 보안 프로토콜이 스마트 오피스 및 스마트 홈과 같은 실제 유비쿼터스 센서 네트워크 환경에 적합하다고 할 수 있다. 본 논문은 1장의 서론에 이어 2장에서는 센서 네트워크의 환경

과 센서 네트워크 보안과 관련된 연구에 대해서 소개 하고 3장에서는 더블키를 이용한 경량 보안 프로토콜에 대하여, 4장에서는 경량 보안 프로토콜 구현, 5장에서는 성능평가, 끝으로 6장에서는 결론 및 향후 연구 방향에 대하여 기술한다.

2. 관련 연구

유비쿼터스 센서 네트워크 환경에서 요구되는 보안 서비스를 만족하기 위해서 먼저 유비쿼터스 센서 네트워크의 환경 구조와 유비쿼터스 센서 네트워크 보안의 목적과 요구되는 보안의 범위에 대하여 명확한 정의가 필요하다. 본 2장에서는 먼저 유비쿼터스 센서 네트워크 환경에 대하여 정의하고 유비쿼터스 센서 네트워크의 구조와 실제 유비쿼터스 센서 네트워크에서 요구되는 보안의 내용 및 본 논문에서 포함하는 보안의 범위에 대하여 설명한다. 그리고 현재 센서 네트워크에서 연구 중인 보안 정책들에서 대하여 기술한다.

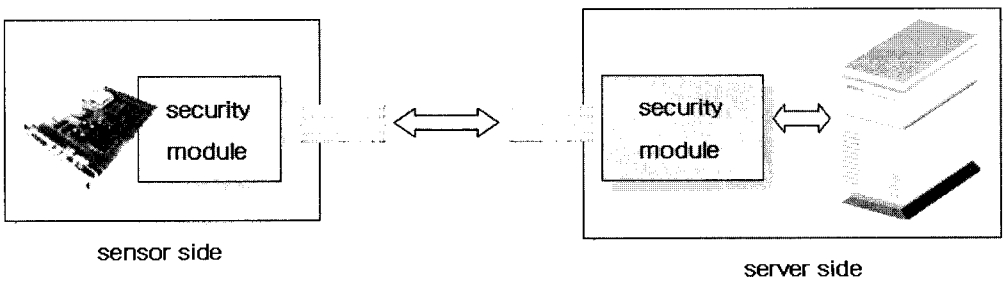
2.1 유비쿼터스 센서 네트워크 환경 및 보안 범위

유비쿼터스 센서 네트워크의 정의 방식에 따라 약간의 차이가 있을 수 있으나 일반적으로 저전력의 많은 센서 노드들이 주변의 환경으로부터 얻어진 정보를 수입하여 인간의 개입 없이 정보를 주고받으면서 데이터를 결정하며 필요한 곳으로 데이터를 전송하고 향후 판단을 위해 데이터를 저장하는 것을 말한다. 유비쿼터스 센서 네트워크와 Ad-hoc 네트워크는 유사하지만 여러 가지 차이점이 있다. 본 논문의 유비쿼터스 센서 네트워크의 보안 적인 측면에서 그 차이점을 기술 하자면 다음과 같다. Ad-hoc 네트워크의 경우 네트워크가 형성 된 후 노드 객체의 네트워크내의 가입 및 탈퇴가 자유로이 할 수 있어야 하며 그렇기 때문에 노드의 이동성을 고려해야 한다. 하지만 센서 네트워크의 경우 각 센서 노드가 임의대로 네트워크 내에 가입 및 탈퇴가 불가능하고 센서 자체의 이동성은 지원하지만 어플리케이션에 의존하지 않는다[14]. 또한 센서 노드는 Ad-hoc 네트워크의 노드들보다 제한된 하드웨어로 구성이 되며 인증 되지 않은 센서의 침입, 위조 혹은 변조 된 데이터를 전달하거나 라우팅 과정에서 정보가 유출되는 것을 막아야 한다.

센서 네트워크의 구조를 유비쿼터스 센서 네트워크 보안 정책을 위한 관점에서 살펴보면 센서 노드와 중앙 관리 서버의 두 가지 객체로 구분할 수 있다(그림 1). 센서 노드는 주변 환경의 데이터를 수집하여 중앙 관리 서버에 전송이 목적인 객체이며 중앙 관리 서버는 센서 노드로부터 온 데이터들을 수집하여 분석, 처리 및 저장이 목적인 객체로 정의 할 수 있다. 중앙 관리 서버의 특징은 다음과 같이 설명할 수 있다. 센서 노드로부터 수집된 데이터를 분석 및 저장을 해야 하며 센서 노드의 수와 센서 네트워크의 구조에 대해서 파악 및 관리를 할 수 있어야 한다. 또한 센서 노드보다 풍부한 하드웨어 자원을 보유하고 있다. 그리고 센서 노드들로부터 보안이 유지된 데이터가 중앙 서버로 전송 되었을 경우에도 보안 처리 관련 모듈만 포함이 된다면 (그림 1)과 같이 서버 측에 어디에 위치를 해도 상관이 없다.

센서 노드의 특징은 다음과 같이 설명할 수 있다. 센서 노드는 중앙 관리 서버와 양방향 통신 혹은 단방향 통신을 한다. 센서 노드는 각각의 고유한 인식자(ID)를 가지고 있으며 한정된 전력과 제한된 하드웨어들로 구성되어 있다. 센서 노드는 해당 센서 네트워크에 임의로 가입 및 탈퇴가 불가능 하다. 센서 노드는 이동성을 보장하지만 임의의 이동이 아닌 중앙 서버가 관리하는 범위 내에서 가능하다. 센서 노드는 중앙 서버하고만 통신이 가능하다. 즉 센서 간의 요청에 의한 통신은 불가능하며 수집된 데이터를 라우팅 하는 연산만 가능하게 한다. 또한 센서 노드 사이에는 전달 받은 데이터를 분석 하거나 재전송을 요구하지 않는다. 이러한 관점에서 보면 센서 노드들의 경우 기존의 기본적인 모듈이외 추가적으로 보안과 관련된 적합한 모듈을 선별적으로 추가해서 제약이 많은 센서 노드에 적용할 수 있는데 이러한 점은 센서의 운영체제나 라우팅 방식, 전송 방식과 무관하게 적용 가능하게 할 수 있다.

일반적으로 보안의 목적은 인증(authentication), 기밀성(confidentiality), 무결성(integrity), 부인봉쇄(non-repudiation)에 기준을 두고 있다. 그 중에 메시지를 전송 했다는 사실을 부인 하지 못하는 부인봉쇄 항목은 센서 노드에서 데이터 전송을 하고 중앙 관리 서버에서 데이터를 받는 유비쿼터스 센서 네트워크의 특징 상 중점을 두지 않아도 될 보안 항목이다. 하지만 인가된 시스템이 자원에 접근할 필요가 있거나 사용하고자 할 때 시스템에 성능에 따라 시스템 자원에 접근 할 수 있도록 하는 가용성(availability) 서비스는 센서 네트워크의 환경에 필요한 보안 항목이다. 센서 네트워크에서의 보안은 암호키를 관리할 수 있는 기능 제공, 센서 환경에 적합한 경량화 된 암호 및 인증 기능 제공, 라우팅



(그림 1) 보안의 객체 개념으로 간략화 된 센서 네트워크의 구조

시에 보안 기능 제공, 서비스 거부 공격에 강한 구조 등을 포함 하고 있어야 한다[2]. 일반적으로 센서 네트워크에서 지원해야 하는 보안 기능으로는 암호 알고리즘, 키 관리 및 보안 프로토콜, 인증 및 시큐어 라우팅, 프라이버시와 침입 시큐어 감시로 나눌 수 있다[4]. 따라서 유비쿼터스 센서 네트워크 환경에서 고려할 보안 항목으로는 인증, 기밀성, 무결성, 가용성이 있으며 보안 요구 사항으로는 다음 <표 1>과 같이 정의 할 수 있다.

〈표 1〉 유비쿼터스 센서 네트워크 환경의 보안 요구 사항

보안 항목	요구 사항 및 보안 범위
인증	• 센서 노드와 중앙 관리 서버와의 인증
기밀성	• 센서 노드에서 보내는 데이터의 기밀성 유지 • 센서 노드 내에서 암호화에 사용하는 키 관리 • 센서 노드 환경에 맞는 경량화 된 암호화 알고리즘
무결성	• 센서 노드에서 보내는 데이터의 무결성 • 공격자로부터 재전송 공격에 대처
가용성	• 센서 네트워크에 센서 노드의 가입 및 탈퇴 처리 • 센서 노드에서 중앙 관리 서버로의 안전한 데이터 라우팅 • 서비스 거부 공격에 대처

<표 1>의 센서 네트워크 환경에서 보안 요구 사항 중에서 제안하는 더블키 기반의 경량 보안 프로토콜의 경우 센서 노드와 중앙 관리 서버간의 인증, 센서 노드 내에서 암호화에 사용하는 키 관리 및 갱신, 공격자로부터 재전송 공격에 대비, 센서 네트워크에 센서 노드의 가입 및 탈퇴 처리에 대하여 보안 범위를 포함하게 된다.

2.3 센서 네트워크 보안 정책

무선 센서 네트워크는 가까운 미래에 널리 사용될 수 있는 기술로서 무선 센서 네트워크 서비스 구성에 대한 연구, 개발이 주류를 이루었으며 보안에 대한 연구는 상대적으로 적은 관심을 보여 왔다. 하지만 점차 센서 네트워크 기반의 서비스에 대한 기술이 구체화 되면서 센서 네트워크상에서 보안에 대한 필요성이 점차 대두되어 보안 기술에 대한 연구가 점차 활발해지고 있다[1].

키 관리 및 접근 제어 등의 암호 서비스를 제공하기 위한 연구로는 센서 노드용 운영체제로 개발된 TinyOS에서 구동 가능하도록 하는 TinySec이 있다[3]. TinySec에서는 링크 보안 기술을 적용한 간단한 접근제어와 메시지 위·변조를 막을 수 있는 무결성 기능과 비합법적인 노드에 의한 센서 정보의 해석을 방지하는 기밀성 기능을 제공한다.

산업체를 중심으로 하는 Zigbee Alliance는 저전력 무선 센서에 다양한 응용의 활용이 가능하도록 하는 기능과 보안 기능이 포함되어 있다[5]. Zigbee 보안 서비스는 대칭키 암호 방식을 이용하여 두 노드 간의 비밀키 설정과 상호 인증 과정을 수행하고, 이 키를 이용하여 MAC계층, 네트워크 계층, 응용 계층에서의 데이터 프레임에 대한 보안 기능을 제공한다. 비밀키의 전달, 상호 인증 등의 프로토콜에 대한 정의는 Zigbee Alliance 규격에 포함되어 있으나, 마스터 비밀키를 안전하게 각 노드에 전달하기 위한 방법을 제공하지는 않는다. 즉, 트러스트 센터라고 불리는 장치를 활용하여 노드 사이의 마스터 비밀키는 중간 노드들의 중계에 의하여 키 전달은 하지만 이 통신 채널의 안전성을 항상 보장하지는 않는다. 대칭키 방식의 암호 메커니즘에서는 비밀키의 안전성을 보장하는 것이 가장 중요함에도 Zigbee 네트워크는 트러스트 센터와 노드 사이의 마스터 비밀키를 공유하고 있다는 가정 하에 암호 통신 서비스를 진행함은 물론, 트러스트 센터에서 통신하고자 하는 모든 노드의 비밀키를 관리하도록 되어 있는 구조적인 약점을 가지고 있다. 새로운 노드의 보안 접속을 위한 비밀키를 분해하고 노드의 해지를 담당하는 트러스트 센터를 어디에 두어야 하느냐에 대한 결정은 Zigbee 표준에서도 이슈로 남아 있는 상태이다.

센서 네트워크 보안 프로토콜로 초기에 발표된 SPINS (Security Protocols for Sensor Networks) 프로토콜은 멀티캐스팅 보안 프로토콜인 TESLA(Timed Efficient Stream Loss-tolerant Authentication)를 간소화시켜 개발한 μ-TESLA와 SNEP (Secure Network Encryption Protocol)로 구성된다. μ-TESLA는 다수로 구성된 센서 네트워크의 기기들의 인증을 담당하며 SNEP는 데이터의 기밀성, 인증을 보장한다[6][7]. μ-TESLA는 두 노드 사이에 공유하여야 하는 비밀키의 노출을 최대한 늦춤으로써 비대칭 암호키 방식을 사용하는 것과 같은 효과를 누릴 수 있다. 하지만 이 방식은 인증하여야 하는 노드 수가 많아질 경우에는 지연 시간이 길어져서 활용이 어렵고 각 노드 간 시간의 싱크 과정도 필요하다는 단점이 있다.

3. 더블키를 이용한 경량 보안 프로토콜

유비쿼터스 센서 네트워크를 위한 경량 보안 프로토콜은 서버와 센서 노드간의 인증 및 키 교환 관리 프로토콜로 이루어져 있다. 더블키를 이용한 보안 프로토콜은 스마트 오피스 및 스마트 홈과 같은 실제 환경의 무선 센서 네트워크에서 각 제약이 많은 센서 노드와 이를 관리 하는 중앙 서버 간의 인증 및 보안 정책에 중점을 두고 설계하였다. 본 논문에서 새롭게 정의하여 사용하는 더블키 정책은 기존의 대칭키 기반의 보안 정책의 단점을 해결하기 위해서 제안되었다. 대칭키 기반의 보안은 대칭키의 분배 문제가 가장 중요한 해결 사항이다. 또한 안전한 통신을 위해서는 주기적으로 혹은 비주기적인 키 교체가 필수적인데 네트워크 환경에서는 구현하기가 쉽지가 않다. 이를 해결하기 위한 방법으로 각각의 다른 객체에서 같은 키를 생성해 내는 연구가 진행 중이지만 이러한 방법을 센서 네트워크의 제한적인 하드웨어 환경의 센서 노드에서 사용하기는 어렵다. 따라서 본 논문에서는 더블키를 이용하여 안전한 인증 뒤 새로운 키 교체 프로토콜을 통해서 주기적으로 키 교체를 할 수 있는 더블키를 이용한 보안 프로토콜을 제안한다.

더블키 정책은 통신을 하는 양쪽의 개체에서 대칭키를 한 개를 소유하는 것이 아닌 두개를 소유하게 한다. 단지 한

개의 대칭키만 저장할 공간만 있다면 가능한 정책으로 분배가 된 두 개의 대칭키 중 하나의 키만 사용을 하고 다른 하나의 대칭키는 사용을 하지 않는다. 두 개의 대칭키 중에서 사용하는 대칭키는 메인키(Main-Key)라고 하며, 사용하지 않지 다른 하나의 대칭키는 보관키(Safety-Key)라고 명칭한다. 키 교체 상황이 되면 기존의 메인키를 지우고 저장하고 있었던 보관키를 사용하는데, 통신을 하는 양쪽 개체에서 저장하고 있던 보관키를 이용하여 새로 생성된 새로운 키를 암호화하여 분배를 하게 된다. 키 교체 프로토콜이 진행 되고 난 후에는 기존의 메인키를 삭제 하고, 저장을 하고 있었던 보관키를 메인키로 사용하고, 새롭게 교환한 키를 보관키로 저장하게 된다. 따라서 각 개체에는 항상 한 개의 대칭키 만으로 암호화 작업을 하지만 두 개의 키를 보관 하게 된다. 인증 프로토콜 진행 시 서버에서 두 개의 대칭키를 생성하여 인증 원하는 센서 노드에 보내게 된다. 첫 번째 보내는 대칭키를 메인키로 사용을 하게 된다. 두 번째 보내는 대칭키는 메인키로 암호화를 하여 전달을 하게 된다. 센서 노드에서는 서버에서 보낸 패킷을 RF(Radio Frequency) 모듈에서 전달 받아서 MCU(Micro Controller Unit)의 레지스터에 저장을 하게 된다. 서버에서 받은 메인키와 보관키는 MCU의 레지스터에서 플래시 메모리로 저장을 하게 된다. 그 중 메인키는 보안 모듈에서 계속해서 사용을 하게 되며, 보관키는 키 교체 프로토콜이 동작하기 전까지는 계속해서 센서 노드의 MCU 플래시 메모리에 보관을 하게 된다. 일반적인 센서 노드에서는 외부에서 플래시 메모리나 SRAM, EEPROM에 접근하는 모듈을 지원하지 않고 있기 때문에 외부에서 보관키의 접근 방법은 없다. 따라서 인증 시 센서 노드에 보관하고 있는 보관키는 안전하다고 할 수 있다.

본 3절에서는 더블키를 이용한 경량 보안 프로토콜에 대하여 설명한다. 3.1절에서는 더블키 보안 프로토콜의 구조, 3.2절에서는 더블키를 이용한 인증 프로토콜, 3.3절에서는 더블키 교체 프로토콜에 대하여 기술한다.

3.1 더블키 보안 프로토콜의 구조

유비쿼터스 센서 네트워크에서 공개키 기반의 보안 정책은 센서 노드의 하드웨어적인 제약 때문에 사용하기가 힘들다. 그렇기 때문에 대부분의 보안 정책은 대칭키 기반의 보안 정책으로 설계한다. 하지만 대칭키 기반의 암호 알고리즘의 경우 마스터키의 분배가 가장 어려운 부분이다. 대부분의 대칭키 기반의 보안 정책에서는 마스터키가 안전한 경로로 분배 되었다고 가정 하에 대칭키 기반의 보안 정책을 설계하기도 하며, 키의 분배가 어렵기 때문에 센서 노드와 관리 서버에서 같은 키를 생성할 수 있는 알고리즘을 사용하여 주기적으로 생성을 해서 교체하는 방식을 제안하기도 한다. 이러한 접근 방식들은 제약이 많은 유비쿼터스 센서 네트워크에서 센서 노드에 실제 적용하기는 힘들다.

본 절에서 이러한 문제점을 해결하기 위해서 더블키를 이용한 보안 정책을 제안한다. 본 논문에서 제안하는 더블키를 이용한 보안 정책은 더블키를 이용한 인증 프로토콜과 더블키 교체 및 관리 프로토콜로 이루어져있다. 제안하는 인증, 키 교체 및 관리 프로토콜은 다음과 같다.

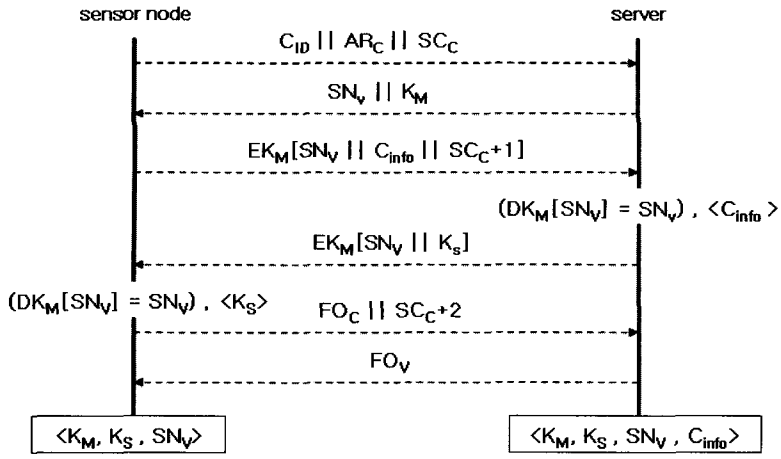
- 인증 프로토콜 : 중앙 관리 서버와 센서 노드간의 인증을 지원하기 위한 프로토콜로써 대칭키를 안전하게 분배 하는 것으로 인증을 하게 된다. 센서 노드는 센서가 초기화가 되고 유비쿼터스 센서 네트워크에서 동작을 하기 전에 중앙 관리 서버와 인증 작업을 한다. 인증이 끝나기 전까지 센서 노드는 어떠한 작업도 시작하지 않는다. 센서 노드와 중앙 관리 서버간의 인증이 완료가 되면 해당 센서 노드는 중앙 관리 서버가 관리하는 유비쿼터스 센서 네트워크에 포함이 되어 정상적인 작동을 시작하게 된다. 인증 과정 중 공격자의 공격으로 판단되거나 네트워크의 이상이 있을 경우 모든 것을 초기화 하고 다시 인증 작업을 시작한다. 서버에서는 인증을 원하는 센서를 관리하기 시작하며 센서 노드에서는 인가된 서버와 대칭키를 공유하고 확인을 함으로써 인증 작업을 끝내게 된다. 인증 프로토콜은 3.2절에서 설명한다.

- 키 교체 및 관리 프로토콜 : 센서 노드와 중앙 관리 서버는 정상적인 인증 작업이 끝나고 나면 각각 두 개의 대칭키를 보관하게 된다. 이는 초기 인증 프로토콜에 의해서 분배가 된다. 그 중 하나의 키는 보관을 하고 하나의 키만으로 암호화 및 복호화에 사용한다. 일정 주기 혹은 중앙 관리 서버의 판단에 의해서 키의 교체가 필요할 경우 센서 노드와 중앙 관리 서버는 기존의 사용하던 마스터키를 삭제하고 새로운 키를 사용하게 되며, 이때 다시 저장용 키를 중앙 관리 서버로부터 받아서 각각 저장하게 된다. 키 교체 및 관리 프로토콜에 대한 설명은 3.3절에서 설명한다.

중앙 관리 서버의 경우 키 생성 알고리즘을 가지고 있으며 또 관리가 가능해야 한다. 또한 키 교체시기를 판단해야 하는 알고리즘을 포함하고 있어야 한다. 센서 노드로부터 오는 데이터를 분석하여 현재의 네트워크 상태 및 센서 노드의 상태를 관리 할 수 있어야 한다. 센서 노드에서는 단지 인증을 받은 후에 서버로부터 받은 키를 보관을 하고 중앙 관리 서버의 신호에 의해서 키를 교체하기만 하면 된다. 이러한 관계는 중앙 관리 서버에서 대부분의 연산과 처리를 담당하게 되고 센서에서는 키 관리 알고리즘만 포함하면 되기 때문에 많은 연산이 필요하지 않게 된다. 기존의 대칭키 보안 정책과 가장 큰 차이점은 센서 노드에 키 생성 및 인증 관리 알고리즘이 포함이 안 되기 때문에 센서 노드에 부하가 적어진다. 또한 안전하게 키 교체를 가능하게 할 수 있다는 장점이 있다.

3.2 더블키를 이용한 인증 프로토콜

유비쿼터스 센서 네트워크의 센서 노드의 환경을 고려해서 센서 노드가 부담해야 하는 처리량을 줄이기 위해 센서



(그림 2) 서버와 센서 노드간의 경량 인증 프로토콜

노드에 몇 가지 사항에 제약이 필요하다. 첫째로는 센서 노드는 인증을 받기 전까지 어떠한 동작도 하지 못하게 해야 한다. 둘째로 센서 노드에 처음 동작을 할 때 전송 거리를 자동으로 최소화 하여 인증을 담당하는 서버와 통신만이 가능하도록 해야 한다. 이러한 초기 제약 조건을 만족 하였을 때 서버와 센서의 인증 프로토콜은 다음 (그림 2)와 같다.

(그림 2)에서 사용 하는 기호의 의미는 다음과 같다.

CID = 센서 ID
|| = 이어 붙여 쓰기
AR_c = 센서 노드에서 보내는 인증 요청 신호
SC_c = 센서 노드에서 생성한 시큐어 카운터
SN_V = 서버에서 설정한 시큐어 넘버
K_M = 첫 번째 대칭키, 메인키
$EK_M[\ ]$ = 메인키를 이용하여 암호화
$DK_M[\ ]$ = 메인키를 이용하여 복호화
C_{info} = 센서의 고유 정보
K_S = 두 번째 대칭키, 보관키
FO_C = 센서 노드에서 보내는 종료 신호
FO_V = 서버에서 보내는 종료 확인 신호
< > = 각 객체에서 저장해야 하는 값

서버와 센서 노드 간의 경량 인증 순서는 다음과 같다.

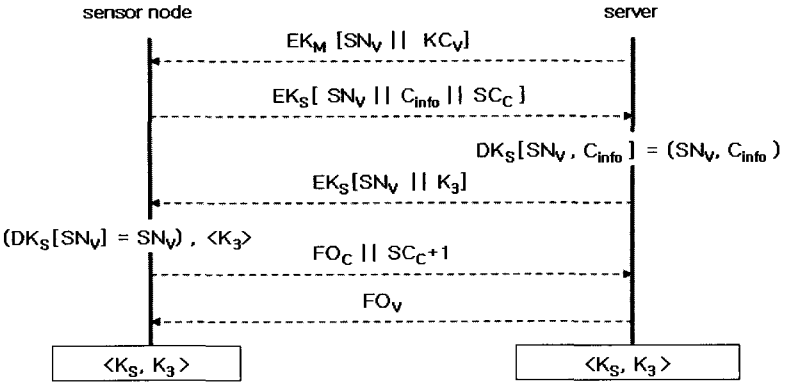
(1) 센서 노드 → 서버 : CID || AR_c || SC_c
(2) 서버 → 센서 노드 : SN_V || K_M
(3) 센서 노드 → 서버 : EK_M [SN_V || C_{info} || SC_c+1]
(4) 서버 : DK_M [SN_V] = SN_V , < C_{info} >
서버 → 센서 노드 : EK_M [SN_V || K_S]
(5) 센서 노드 : DK_M [SN_V] = SN_V , < K_S >
센서 노드 → 서버 : FO_C || SC_c+2
(6) 서버 → 센서 노드 : FO_V

먼저 센서 노드가 하드웨어 초기화를 하고 난 뒤 동작을 시작하게 되면 먼저 센서 노드에서 주변의 인증 처리 서버에 인증 요청하는 신호를 보내게 된다. 센서 노드가 서버의 주변에서 무선으로 혹은 유선으로 전송하도록 하여 공격자의 공격에 대비하도록 한다. 그러한 상황은 공격자가 센서 노드의 해당 신호를 받아서 인증 서버 대신해서 처리를 할 수가 없으며 서버에서는 인증을 해야 하는 센서의 정보를 가지고 있기 때문에 공격자의 센서가 인증을 요청 하여도 서버에서 확인이 가능하게 된다. 또한 공격자가 하나의 센서 노드를 차단하고 해당 센서의 정보를 이용하여 인증을 대신 받더라 하더라도 서버와 센서 노드 간의 인증만 이루어 질 뿐 어떠한 정보를 교환 하거나 네트워크에서 사용하는 키를 교환 하는 것이 아니기 때문에 다른 센서에서 사용되는 대칭키가 전복될 수는 없다.

두 번째 단계에서 인증 처리 서버에서는 인증 요청 신호가 오게 되면 시큐어 넘버 (Secure Number)를 생성하고 서버에서 해당 센서 노드의 환경에 맞는 키를 생성하게 된다. 이렇게 생성된 키와 시큐어 넘버를 요청을 한 센서로 전달을 하게 된다. 서버에서 센서로 가는 전송 역시 전송 거리에 제한이 있기 때문에 공격자가 이것을 복사하기 위해서는 여러 가지 문제점을 해결해야만 가능하다. 서버에서는 인증을 해야 하는 센서 노드의 하드웨어 정보를 저장하고 있기 때문에 다른 공격자의 센서 노드가 대신하여 인증 프로토콜을 완료 하기는 어렵다. 만일 공격자가 대칭키를 복사 하더라도 해당 센서 노드와 서버간의 대칭키이기 때문에 다른 센서 노드가 전복될 위험은 없게 된다.

세 번째 단계에서는 센서 노드에서는 서버에서 보낸 시큐어 넘버와 자신의 센서 정보에 대해서 서버에서 받은 키를 이용해서 암호화를 한 뒤 다시 서버로 전송 한다. 이 과정에서 공격자가 복사한 키를 이용하여 자신의 센서 정보를 암호화 하였을 경우 서버 쪽에서는 두 개의 응답을 받기 때문에 인증이 무효화가 되며 센서의 정보가 암호화가 되어서 전달되기 때문에 어떠한 센서가 사용하는 키인지 확인이 불가능하게 된다.

네 번째 단계에서는 서버에서는 암호화로 받은 문장을 복호화를 해서 시큐어 넘버를 비교하게 된다. 다른 키를 이용



(그림 3) 센서 노드와 서버간의 키 교체 프로토콜

하거나 시큐어 넘버 값이 틀릴 경우 인증은 무효화가 되고 일치 할 경우에는 센서 정보와 시큐어 넘버 그리고 교환한 키를 저장하게 된다. 저장이 끝난 뒤 서버에서는 두 번째 키를 생성하여 시큐어 넘버와 두 번째 키를 첫 번째 키로 암호화를 해서 다시 센서에 전송을 하게 된다.

다섯 번째 단계에서는 센서 노드에서는 첫 번째 암호키로 복호화 작업을 하고 복호화 작업을 해서 얻은 시큐어 넘버와 저장을 해둔 시큐어 넘버를 확인하고 일치 할 경우 두 번째 마스터키를 저장을 한다. 작업이 끝나면 인증 서버에 종료 신호를 보내다.

여섯 번째 단계에서는 서버에서는 이 종료신호를 받고 확인 신호를 보내면 인증 작업을 끝내게 된다. 인증 알고리즘에 의해서 인증이 끝나고 나면 센서에는 두 가지 마스터키와 시큐어 넘버를 저장하게 되고 관리 하게 된다. 인증 서버에서는 두 가지 대칭 마스터키와 시큐어 넘버 그리고 해당 센서의 센서 정보를 저장한다.

3.3 더블키 교체 및 관리 프로토콜

센서 네트워크에서 하나의 대칭키를 이용 할 경우 위험 부담은 증가하게 된다. 안전한 방식에 의해서 키가 분배가 되었다고 하더라도 키가 노출이 된다면 네트워크 전체에 영향을 주게 된다. 이런 점을 방지하기 위해서는 네트워크 내에서 여러 개의 키를 사용하거나 키를 계속 바꿔주는 알고리즘이 필수적이다. 하지만 센서 노드의 제약은 센서 노드 안에 서버와 같은 키를 생성하게 하는 알고리즘을 포함시키기 어렵고 또 각 센서마다 다른 알고리즘을 사용해서 각각 서버와 같은 키를 공유해야 하는데 센서 마다 다른 알고리즘을 주거나 랜덤 값을 이용해서 공유하기는 쉽지가 않다. 따라서 센서 노드의 하드웨어적인 환경을 고려한 서버와 센서 노드간의 주기적인 키 교체 알고리즘과 안전하게 대칭키를 공유하는 알고리즘이 절실히 필요하다.

본 절에서는 유비쿼터스 센서 네트워크 보안 관리자의 판단이나 관리자가 정해 놓은 키 교체 주기 마다 센서 노드와 서버간의 안전한 키 교체 알고리즘을 제안한다. (그림 3)은 서버와 센서 노드간의 키 교환 프로토콜을 나타낸 것이다. 교체 주기는 보안 정책에 맞도록 기간을 정할 수가 있으며 서버의 보안 모듈의 판단으로 키 교체가 가능하도록 설계하였다.

(그림 3)에서 사용하는 기호의 의미는 다음과 같다.

C_{ID} = 센서 ID
KC_V = 서버에서 센서 노드에서 보내는 키 교체 신호
|| = 이어 붙여 쓰기
SC_C = 센서 노드에서 생성한 시큐어 카운터
SN_V = 서버에서 설정한 시큐어 넘버
K_S = 저장중인 대칭키, 보관키
EK_M[ ] = 기존에 사용 중인 메인키를 이용하여 암호화
EK_S[ ] = 저장중인 보관키를 이용하여 암호화
DK_S[ ] = 저장중인 보관키를 이용하여 복호화
C_{info} = 센서의 고유 정보
K_3 = 새로운 대칭키
FO_C = 센서 노드에서 보내는 종료 신호
FO_V = 서버에서 보내는 종료 확인 신호
< > = 각 객체에서 저장해야 하는 값

서버와 센서 노드 간의 키 교체 관리 순서는 다음과 같다.

(1) 서버 → 센서 노드 : EK_M [SN_V || KC_V]
(2) 센서 노드 → 서버 : EK_S [SN_V || C_{info} || SC_C]
(3) 서버 : DK_S [SN_V || C_{info}] = (SN_V , C_{info})
서버 → 센서 노드 : EK_S [SN_V || K_3]
(4) 센서 노드 : DK_S [SN_V] = SN_V , < K_3 >
센서 노드 → 서버 : FO_C || SC_C+1
(5) 서버 → 센서 노드 : FO_V

첫 번째 단계에서 서버의 보안 모듈이나 관리자에 의해서 키 교체 프로토콜이 실행이 되면 먼저 서버에서 해당 센서 노드에 키 교체를 통보하는 신호를 보내게 된다. 이 신호는 기존에 사용하던 메인키를 이용하여 전송을 하게 된다.

두 번째 단계에서는 센서 노드에서 키 교체 신호를 받게 될 경우 사용 중인 메인키가 아닌 저장을 해 둔 보관키를 이용해서 정보를 서버로 보내게 된다. 만일 공격자에서 의해서 키 교체를 통보하는 신호를 센서가 받더라도 공격자는

저장 되어 있는 키를 모르기 때문에 서버를 대신 해서 응답을 받더라도 재 응답을 할 수가 없게 된다. 서버의 경우 키 교체 신호를 보내지 않았는데도 불구하고 응답이 올 경우에는 공격자에 의해서 네트워크가 공격되고 있다는 가정으로 관찰을 시작하고 해당 센서를 감시 혹은 네트워크에서 일시 제외 등의 조취를 취할 수 있다.

세 번째 단계에서 정상적으로 서버에서 센서 노드의 응답을 받게 되면 센서 노드가 보낸 정보를 저장이 되어 있던 키로 복호화를 해서 데이터 값을 맞추어 보게 된다. 이 경우 이미 저장 되어 있던 키를 사용하기 때문에 인증 된 서버와 센서 노드 간의 키 교체를 위한 인증을 할 수 있게 된다. 서버에서 센서 노드와 키 교체 상태가 되면 새로운 키를 다시 저장되어 있던 키를 통해서 암호화를 하고 다시 센서에 전송하게 된다. 이 경우 해당 센서가 아닌 경우에는 키 값이 존재 하지 않기 때문에 의미가 없고 센서 노드의 경우에도 이미 저장이 되어 있는 안전한 키를 통해서 데이터를 복호화하고 비교를 하기 때문에 안전하게 분배가 가능해 진다.

네 번째 단계에서 센서 노드에서 받은 정보가 일치 하게 되면 기존에 사용하던 키(K_M)는 삭제를 하고 저장 해 두었던 키(K_S)를 메인키로 사용하게 되며($K_S \rightarrow K_M$) 새로 받은 키(K_3)를 다시 보관키로 저장($K_3 \rightarrow K_S$)하게 된다. 마찬가지로 서버에서도 센서 노드로부터 프로토콜 종료 신호가 오게 되면 새로운 값을 저장하고 기존의 암호키는 삭제를 하게 된다.

새로운 키 교체 시기는 키의 크기에 따라 틀려지며 시스템 관리자의 판단과 네트워크의 상태에 따라 바뀌어 지게 된다. 본 절에서 제안하는 대칭키 교체 프로토콜의 경우 교체 시간의 제한 및 알고리즘 내내 데이터를 비교하면서 교체를 하기 때문에 중간에 공격자의 공격을 쉽게 알아 낼 수 있게 되어 있다. 제안된 프로토콜은 센서 노드의 하드웨어 환경에 맞도록 설계되었기 때문에 센서 노드에 적은 부하로 안전하게 대칭키를 교환 할 수 있는 적합한 알고리즘이라고 할 수 있다.

4. 더블키를 이용한 경량 보안 프로토콜 구현

유비쿼터스 센서 네트워크 보안 정책에서 보안 객체는 앞선 (그림 1)과 센서 노드와 서버 노드의 두 가지 객체로 구분할 수 있다. 센서 노드는 주변 환경의 데이터를 수집하여 중앙 처리 서버에 전송이 목적인 객체이며 하며 중앙 처리 서버에서는 센서 노드로부터 온 데이터들을 수집하여 분석, 처리 및 저장이 목적인 객체로 구분 가능하다. 이러한 관점에서 보면 센서 노드들의 경우 기존의 기본적인 모듈이외 추가적으로 보안과 관련된 적합한 모듈을 선별적으로 추가해서 제약이 많은 센서 노드에 적용할 수 있는데 이러한 점은 센서의 운영체제나 라우팅 방식, 전송 방식과 무관하게 적용 가능하게 할 수 있다. 또한 센서 노드들로부터 보안이 유지된 데이터가 중앙 서버로 전송 되었을 경우에도 보안 처리 관련 모듈만 포함이 된다면 (그림 1)에서처럼 서버 측에 어디에 위치를 해도 상관이 없다. 본 논문에서 제안한 더블키를 이용한 보안 프로토콜을 구현하기 위해서는 서버 측과 센서 노드 측에 보안 프로토콜을 처리하기 위한 보안 컴포넌트들이 필요하다. 4.1절에서 더블키 보안 프로토콜이 동작되기 위하여 서버와 센서 노드의 보안 컴포넌트 구조에 대해여 설명하고 4.2절에서는 구현된 경량 보안 프로토콜에 대하여 설명한다.

4.1 경량 보안 프로토콜을 위한 보안 컴포넌트 구현

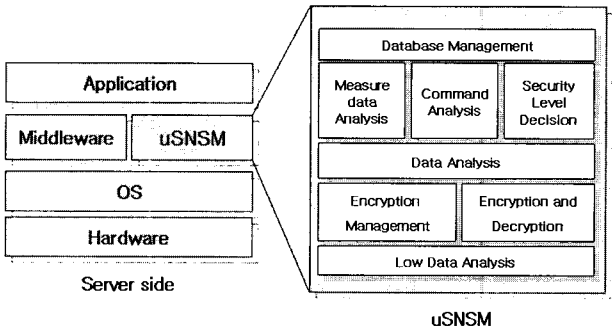
경량 보안 프로토콜을 위한 보안 컴포넌트는 서버 측에 포함이 되는 보안 컴포넌트와 센서 노드 측에 포함되는 보안 컴포넌트로 이루어져 있다. 본 논문에서는 서버 측의 보안 컴포넌트는 유비쿼터스 센서 네트워크 보안 관리자(Ubiquitous Sensor Network Security Manager)라는 명칭으로 정의해서 사용하며, 센서 측의 보안 컴포넌트는 경량 보안 관리자(Light-Weight Security Manager)이라는 명칭으로 정의해서 사용한다. 본 4.1.1 절에서는 서버 측의 보안 컴포넌트인 유비쿼터스 센서 네트워크 보안 관리자(uSNSM)에 대한 구조 및 각각의 모듈에 대해서 설명하며 uSNSM에서 생성하는 패킷의 구조에 대하여 설명한다. 4.1.2 절에서는 센서 노드 측의 보안 컴포넌트인 경량 보안 관리자(LWSM)에 대한 구조 및 모듈에 대하여 설명하고 LWSM에서 생성하는 패킷의 구조에 대하여 설명한다.

4.1.1 서버를 위한 유비쿼터스 센서 네트워크 보안 관리자

서버에 동작이 되는 uSNSM는 Windows 환경에서 동작이 되며, 프로그래밍 언어로는 C++를 이용하였다. (그림 4)의 모듈 중에서 로우 데이터 분석 모듈 (Low Data Analysis module)과 데이터 저장 관리 모듈 (Database Management module)은 기존의 데이터 처리 모듈을 사용 할 수도 있으며 독립적으로 동작을 할 수 있도록 설계되어서 재사용성을 고려하였다. 유비쿼터스 센서 네트워크 보안 관리자에서 각 모듈의 특징은 다음과 같다.

• 로우 데이터 분석 모듈 (Low Data Analysis module)

로우 데이터 분석 모듈에서는 센서 노드에서 전송된 암호화가 된 로우 데이터를 복호화 하기 전에 어떠한 방식으로 처리를 할지 결정을 하게 된다. 센서 노드에서 전송된 로우 데이터라고 하더라도 암호화가 된 부분과 암호화가 되지 않



(그림 4) 유비쿼터스 센서 네트워크 보안 관리자(uSNSM) 구조

은 부분이 있기 때문에 암호화가 되지 않은 부분에서 필요한 데이터를 추출해서 암호화 된 데이터 부분을 처리 할 수 있도록 한다. 이 부분은 보안 모듈이 없더라도 서버 쪽에서 로우데이터를 받으면 처리하는 모듈이기 때문에 기존의 모듈을 이용할 수 있으며 보안 처리 결정만 포함하면 되므로 굳이 새로 설계를 할 필요는 없으며 기존의 모듈을 수정하여 사용가능한 부분이다.

• 암호화 및 복호화 모듈 (Encryption and Decryption module)
암호화 및 복호화 모듈은 로우 데이터의 분석이 끝나고 암호화와 복호화를 담당하는 모듈로서 차후 여러 암호화 알고리즘과 암호키를 사용할 수 있도록 확장이 가능하다.

• 센서 노드 암호화 관리 모듈 (Encryption Management module)
센서 노드 암호화 관리 모듈에서는 센서에서 사용하게 되어질 대칭키를 생성하고 관리 하는 모듈이다. 또한 보안 요소로 사용하고 있는 시큐어 넘버를 생성을 하는 모듈이다. 그리고 새로운 센서의 등록 및 탈퇴를 할 때 센서 노드 암호화 관리 모듈을 통하여 처리를 하게 된다.

• 데이터 분석 모듈 (Data Analysis module)
데이터 분석 모듈에서는 복호화 작업이 끝난 데이터를 분석하는 모듈이다. 이 모듈에서 센서 노드로부터 온 데이터의 종류 및 처리 방향을 결정하게 된다.

• 계측 데이터 처리 모듈 (Measure data Analysis module)
센서 노드로부터 전송된 데이터가 해당 센서 노드에서 측량한 데이터일 경우에 처리하는 모듈이다. 먼저 암호화 된 데이터와 암호화가 되지 않은 부분의 센서 노드 정보를 비교해서 무결성을 확인하게 된다. 또한 센서 노드에서 보낸 시큐어 카운터(Secure Counter)를 이미 해당 노드에서 받은 값들과 비교하여 무결성, 가용성을 확인하게 된다. 이때 유비쿼터스 센서 네트워크 보안 관리자에서 공격자의 공격이 의심 되면 보안 레벨 결정 모듈에 신호를 보내서 공격자에 대해서 능동적으로 대처를 하도록 한다. 또한 uSNSM과 센서 노드의 보안 레벨을 조정하도록 한다.

• 명령 데이터 처리 모듈 (Command Analysis module)
센서 노드로부터 전송된 데이터가 계측 데이터가 아닌 서버와 센서 노드간의 인증 혹은 센서의 처리 요구 데이터로 분석이 되면 명령 데이터 처리 모듈이 실행이 된다.

• 보안 레벨 결정 모듈 (Security Level Decision module)
보안 레벨 결정 모듈에서는 센서 노드로부터 온 데이터 분석 후에 공격자로부터 공격에 대처 하거나 관리자가 판단에 의해서 보안의 레벨을 조정해야 하는 상황에서 실행되는 모듈이다. 보안 레벨은 센서 노드의 전력과도 밀접한 관계가 있는 모듈이다.

• 데이터 저 장 관리 모듈 (Database Management module)
데이터 저장 관리 모듈은 센서 노드로부터 온 데이터를 저장하거나 다른 응용 프로그램에서 사용할 수 있도록 처리하는 모듈이다. 또한 데이터의 분석을 위해서 다른 미들웨어나 처리 장치로 보내는 역할을 하기 때문에 기존의 중앙 관리 서버에서의 모듈과 비슷한 동작을 하는 모듈이 있다면 확장이 가능하고 또 재사용이 가능한 모듈이다.

본 논문의 서버 측의 uSNSM에서 센서 노드로 가는 패킷의 구조를 (그림 5)와 같이 정의 하였다. 서버에서 센서 보내는 데이터 패킷은 센서와 인증 및 키 교환 할 때, 그리고 센서 노드의 보안 레벨을 변경하여 센서 노드의 상태를 변경할 때 전송하게 된다.

(그림 5)에서 Data type과 SL은 서버의 uSNSM에서 센서로 보내는 데이터의 종류 및 보안 관련 명령을 구별하기 위해 사용하는 부분이다. 인증, 키 교환과 같은 데이터 교환 및 센서의 전송 거리 변경 및 전송 간격 변경을 처리하기 위한 명령어를 나타낸다. Node ID, Packet Type, Length 부분을 제외한 나머지 부분은 필요에 따라 암호화가 되어서 전송이 되는 부분이다.

4.1.2 센서 노드를 위한 경량 보안 관리자

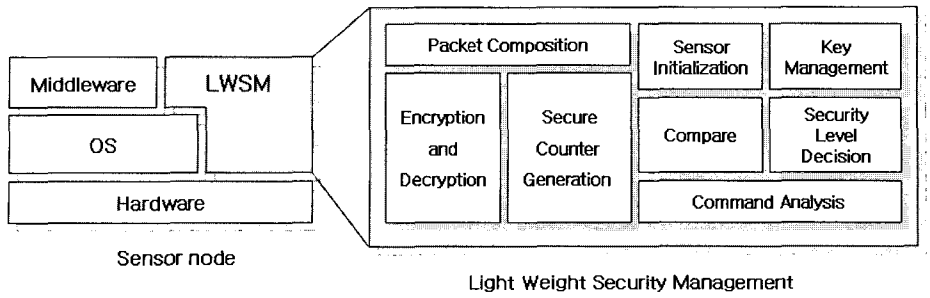
센서 노드에 포함 되는 경량 보안 관리자의 구조는 (그림 6)과 같다. 경량 보안 관리자는 센서 노드의 제한적인 하드웨어

1 byte	1 byte	1 byte	1 byte	1 byte	4 byte	1 byte
Node ID	Packet Type	Length	Data type	S L	SN	Padding

1 byte	1 byte	1 byte	1 byte	1 byte	4 byte	7 byte	1 byte
Node ID	Packet Type	Length	Data type	S L	SN	Key	Padding

Data type : 데이터 타입
SL : 보안 관련 명령어
Padding : 패딩
SN : 시큐어 넘버 정보
Key : 암호키

(그림 5) uSNSM에서 센서 노드로 보내는 패킷의 구조



(그림 6) 센서 노드의 경량 보안 관리자 구조

특성을 고려해서 설계되었으며 운영체제나 전송 방식에 독립적으로 동작하도록 설계하였기 때문에 확장성이 가능하다. 센서 노드에 포함이 되는 경량 보안 관리자의 각 모듈은 다음과 같은 보안 처리를 담당하게 된다.

• 패킷 구성 모듈 (Packet Composition module)

센서 하드웨어가 주변 환경에 대한 데이터를 수집하였을 경우 그 데이터를 전송이 가능한 패킷으로 구성하는 모듈이다. 패킷의 구성은 센서 노드의 운영체제에 종속적이기 때문에 다른 운영체제를 가진 센서 노드들은 다른 패킷으로 구성 될 수밖에 없지만 본 논문에서는 암호화 부분과 비 암호화 부분으로 구분하여 패킷을 구성하기 때문에 본 논문에서 정의된 부분만 포함을 하게 된다면 어떠한 방식의 패킷 구성 모듈이라도 사용할 수 있게 설계 되었다.

• 암호화 모듈 (Encryption (and Decryption) module)

센서 노드에서 전송하는 데이터를 암호화를 하거나 복호화를 하도록 처리하는 모듈이다. 실제 패킷 구성 모듈에서 전송 될 패킷이 완성이 되면 암호화 부분과 비 암호화 부분으로 나누어서 처리를 하게 된다. 암호화 모듈은 센서 노드가 초기화가 될 때 직접적으로 관리자에 의해서 결정이 되지만 센서 노드가 하드웨어 적으로 발전을 하였을 경우에 암호화 모듈을 관리자에 의해 센서의 동작 중에 교체 할 수 있도록 확장 가능하도록 설계 되었다.

• 명령어 처리 모듈 (Command Analysis module)

명령어 처리 모듈은 서버로부터 온 명령어를 분석을 하는 모듈이다. 서버로부터 온 데이터에 대해여 어떤 동작을 해야 할지를 결정해서 다른 모듈을 호출하게 된다.

• 센서 초기화 모듈 (Sensor Initialization module)

센서가 처음 부팅이 되고 인증 및 초기화를 위한 모듈이다. 보안 상태를 위지하기 위한 초기화 모듈이 포함되어 있다.

• 키 관리 모듈 (Key Management module)

센서 노드에서 사용하는 키를 저장 및 관리하는 모듈이다.

• 분석 모듈 (Compare module)

서버로부터 온 명령어에 따라 센서 노드 내의 데이터를 분석하는 모듈이다. 가용성을 지원하기 위해서 서버로부터 오는 데이터를 확인하고 또한 이상 여부를 결정하는 모듈이다.

• 보안 레벨 결정 모듈 (Security Level Decision module)

서버로부터 혹은 관리자의 요구에 의해서 보안 레벨을 조정하도록 하는 모듈이다. 알고리즘, 암호키 교환, 전송 거리 교체 등을 처리 할 수 있는 모듈이며 센서 노드의 전원 이상이나 센서 노드의 네트워크 내에 탈퇴 여부를 결정 할 수 있게 하는 부분이다.

• 시큐어 카운터 생성 모듈 (Secure Counter Generation module)

센서 노드에서 전송하는 데이터의 무결성을 보장하기 위해서 센서 노드 내에서 시큐어 카운터를 생성하여 전송 데이터에 포함 시킬 수 있는 모듈이다.

센서 노드를 위한 경량 보안 관리자를 구현하기 위한 센서 노드의 하드웨어 사양은 <표 2>와 같다. 센서 노드에 동작이 되는 경량 보안 관리자는 C 언어로 구현 하였다.

〈표 2〉 센서 노드의 하드웨어 사양

항목	사양
CPU	ATmega128L
flash memory (KB)	512 KB
EEPROM	4 kB
SRAM	32 kB
RF	CC2420(2.4GHz,IEEE802.15.4, 250kbps Effective Data rate)
System Clock	8 MHz
RF Clock	16 MHz
Operation System	Qplus
Sensor	온도,습도,조도,가스,초전형,초음파
Power	3V AA battery 2EA

또한 센서 노드에서 전송 하는 패킷의 구조는 (그림 7)과 같다. (그림 7)에서 Node ID, Packet Type, Length 부분은 Qplus에서 지정한 패킷의 구조이다. 그리고 Data type, Time Count, S_{INFO}, Time Stamp 등의 부분은 새로 정의된 보안 관련 구조이다. 실제 Node ID, Packet Type, Length 부분을 제외한 나머지 부분만 암호화 처리가 된다. 본 논문

1 byte	1 byte	1 byte	1 byte	1 byte	1 byte	1 byte	2 byte	1 byte
Node ID	Packet Type	Length	Data type	Sensor ID	ADCH	ADCL	Secure counter	Padding

1 byte	1 byte	1 byte	1 byte	1 byte	2 byte	2 byte	1 byte
Node ID	Packet Type	Length	Data type	Sensor ID	data	Secure counter	Padding

1 byte	1 byte	1 byte	1 byte	1 byte	4 byte	2 byte	2 byte	4 byte
Node ID	Packet Type	Length	Data type	Sensor ID	Secure Number	C_{INFO}	Secure counter	Padding

Node ID : Sensor 노드별 ID
Packet Type : 패킷의 타입
Length : 데이터에 속한 센서의 수
Data type : 데이터 타입
Sensor ID : 센서 지정 ID
Secure Number: 시큐어 넘버
C_{INFO} : 센서 정보
Secure counter: 센서 시큐어 카운터
data : 요구 데이터
Padding : 패딩

(그림 7) 센서 노드에서 전송되는 패킷의 구조

에서 제안하는 유비쿼터스 센서 네트워크에서 더블키를 이용한 경량 보안 프로토콜의 센서 노드에서 보안 정책이 포함된 패킷은 세 가지로 구분이 된다. 그중 센서 노드에서 주변 환경 데이터를 수집해서 보내는 기본 패킷은 7 byte이며, 보안 정책이 적용된 후에는 10 byte가 된다. 그 외 구조는 인증, 키 교체 그리고 센서 노드의 전력량 등의 값과 같은 서버에서 요구한 데이터 전송과 같이 보안 전용으로 사용하게 된다. 그 크기는 10 byte, 17 byte의 두 가지 크기로 정해져있다.

4.2 더블키를 이용한 인증 프로토콜 및 키 교체 프로토콜 구현

유비쿼터스 센서 네트워크에서 더블키를 이용한 인증 프로토콜은 (그림 8)과 같다. [(그림 8)에서 왼쪽의 프로토콜은 센서 노드에서 처리하는 프로토콜이며 C로 프로그래밍 되어있다. 오른쪽의 프로토콜은 서버에서 처리하는 프로토콜이며 C++로 프로그래밍 되어있다. 두 가지의 프로토콜이 서로 데이터를 주고받으면서 대칭키를 분배하며 인증을 완료하게 된다.

(그림 8)에서 왼쪽의 센서 노드에서 동작하는 프로토콜의 첫줄에서 서버로부터 온 데이터 중 Data type과 SL 부분을 해석한다. 분석 결과 서버에서 보낸 메시지의 종류에 따라서 세 가지 종류로 구분하여 동작한다. 또한 서버의 인증 프로토콜도 센서 노드로부터 전달된 데이터를 분석하여 세 가지로 구분하여 인증 프로토콜이 동작을 하게 된다. 서버 노드의 세 번째 줄에서는 센서 노드로부터 온 인증 요청 데이터를 분석하여 센서 정보를 확인하게 된다. 만일 이미 인증이 끝난 센서에서 인증 요청이 올 경우 보안 감시자 모듈로 신호를 보내게 된다. 인증 요청이 처음 일 경우 시큐어 넘버와 대칭키 2개를 생성하여 그 중 시큐어 넘버와 하나의 키를 센서로 보내게 된다. 센서 노드의 인증 프로토콜에서 세 번째 줄의 경우 서버로부터 시큐어 넘버와 첫 번째 대칭

```
1  analyze DATA TYPE, SL
2  switch ( SL )
3      case 0 : save SecureNumber, K1 (temporarily)
4               encrypt SecureNumber, Sinfo to K1
5               send SecureNumber Sinfo to server
6      case 1 : if (compare SecureNumber)
7                  save K2 (temporarily)
8                  send finish message to server
9              else send the warning message
10     case 2 : save SecureNumber,K1,K2
```

```
1  analyze DATA TYPE
2  switch (DATA TYPE)
3      case 0 : if (exist Sensor ID) send the warning message
4               generate SecureNumber, K1, K2
5               send SecureNumber, K1
6      case 1 : if (compare SecureNumber)
7                  save Sinfo (temporarily)
8                  encrypt SecureNumber, K2 to k1
9                  send SecureNumber, K2 to sensor
10             else send the warning message
11     case 2 : save SecureNumber, K1, K2, Sinfo
12              send finish message to sensor
```

(그림 8) 서버와 센서 노드의 더블키 인증 프로토콜

```
1 analyze DATA TYPE, SL
2 switch ( SL )
3     case 3 :  encrypt SecureNumber, Sinfo to K1
4               send SecureNumber Sinfo to server
5     case 4 : if (compare SecureNumber)
6                save K3 (temporarily)
7                send finish message to server
8             else send the warning message
9     case 5 : save K2, K3
```

```
1 analyze DATA TYPE
2 switch (DATA TYPE)
3     case 3 : if (compare SecureNumber, Sinfo)
4                generate k3
5                save Sinfo (temporarily)
6                encrypt SecureNumber, K3 to k1
7                send SecureNumber, K3 to sensor
8            else send the warning message
9     case 4 : save K2, K3
10           send finish message to sensor
```

(그림 9) 서버와 센서 노드의 더블키 교체 프로토콜

키를 받고 첫 번째 대칭키를 이용하여 데이터를 암호화 하여 서버로 전송하게 된다. 서버 인증 프로토콜의 여섯 번째 줄에서 센서 노드로부터 온 데이터를 분석하여 처음 보낸 시큐어 넘버를 확인하고 두 번째 대칭키를 암호화 하여 센서 노드로 보낸다. 센서 노드 인증 프로토콜의 여섯 번째 줄에서 인가된 서버로부터 온 데이터인지를 확인한 후 만일 데이터의 값이 다를 경우 서버로 경고 메시지를 전송하게 된다. 서버 인증 프로토콜의 열한 번째 줄과 센서 노드 인증 프로토콜의 열 번째 줄에서 서로의 종료 확인 메시지가 오면 각각 시큐어 넘버, 첫 번째와 두 번째 대칭키, 그리고 서버에선 해당 센서의 정보까지 저장하게 된다.

대칭키 기반의 보안 정책의 경우는 키 분배가 가장 중요한 사항이다. 본 논문에서는 더블키를 이용하여 안전하게 키를 교체하도록 한다. 유비쿼터스 센서 네트워크에서 더블키 교체 및 관리 프로토콜은 (그림 9)와 같다.

(그림 9)에서 왼쪽의 프로토콜은 센서 노드에 포함되어 있는 더블키 교체 프로토콜이며 오른쪽의 프로토콜은 서버에 더블키 교체 프로토콜이다. 서버로부터 대칭키 교체 신호가 오면 센서 노드 교체 프로토콜의 세 번째 줄에서 기존의 사용하는 대칭키로 데이터를 암호화 하여 전송하게 된다. 전송된 데이터는 서버 교체 프로토콜의 세 번째 줄에서 데이터를 비교하게 된다. 교체 신호를 보낸 적이 없는데 데이터가 왔거나 데이터가 다를 경우 보안 관리자 모듈에 경고 메시지를 전송하게 된다. 만일 데이터가 맞을 경우 센서 노드에서 데이터를 받을 준비가 되었음으로 판단하여 네 번째 줄에서 새로운 대칭키를 생성하여 여섯 번째 줄에서처럼 기존의 대칭키로 새로운 대칭키를 암호화 하여 전송하게 된다. 센서 노드에서는 다섯 번째 줄에서처럼 데이터 확인한 후 서버로 종료 메시지를 보내게 된다. 서버의 교체 프로토콜의 아홉 번째 줄과 센서 노드의 교체 프로토콜의 아홉 번째 줄에서 각각의 데이터를 저장하고 프로토콜을 종료하게 된다.

5. 성능 평가

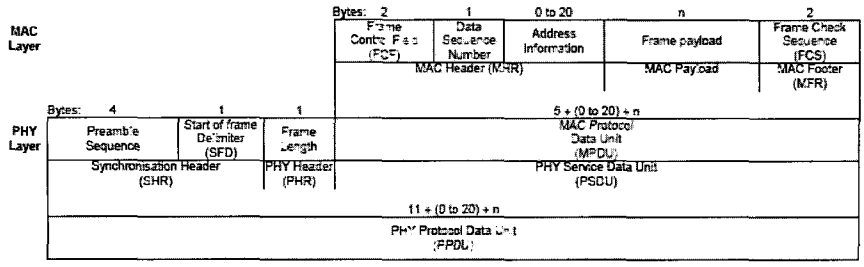
센서 노드의 전력은 약 60%가 데이터를 전송하고 수신할 때 소비가 되며 약 40%가 센서 노드의 동작에 소비가 된다. 따라서 제안한 보안 프로토콜을 위한 성능평가 항목으로 보안 정책이 포함되기 전후의 전송 되는 패킷량의 증가 및 패킷 크기의 증가율이 성능 평가에 중요한 항목이 될 것이다. 또한 센서 노드의 동작에서 전력의 소비가 이루어지기 때문에 센서 노드에 추가적으로 포함되는 모듈의 크기도 성능 평가에 중요한 항목이 될 것이다. 따라서 본 논문에서 제안한 유비쿼터스 센서 네트워크를 위한 경량 보안 프로토콜의 타당성을 검증하기 위해서 다음과 같은 성능평가 항목을 설정하였다.

- 보안 정책 적용 후 패킷 크기 증가율
- 인증, 키 교체 알고리즘 실행 시 추가적인 패킷 전송량 증가에 따른 전력 소비
- 센서 노드에 포함되는 대칭키 기반의 보안 모듈과 일반 보안 정책의 모듈 크기 비교

센서 노드에서 전력의 소비를 줄이기 위한 방안은 크게 전송 되는 패킷의 양을 줄이거나 센서 노드에서 일정 동작을 하지 않을 경우 휴식 상태가 되게 하여 전력의 소비를 줄이는 방안들이 연구 중에 있다. 하지만 보안 정책을 포함된 센서 노드는 추가적인 센서 노드의 동작을 발생 시키고 패킷의 크기 및 전손 되는 패킷의 양이 늘어나기 때문에 전력 소비 증가는 피할 수 없을 것이다. 따라서 성능 평가에서 중점적으로 볼 것은 일반적인 상태에서와 기존의 보안 정책이 포함되었을 경우, 그리고 본 논문에서 제안하는 유비쿼터스 센서 네트워크에서 더블키를 이용한 경량 보안 프로토콜의 보안 정책이 포함되었을 경우의 센서 노드에 적재가 되는 프로그램의 크기를 비교하여 소비 되는 전력의 상대적인 차이를 측정하는 것이다.

5.1 보안 정책 적용 후 패킷 크기 증가율

패킷의 크기는 보안 정책이 포함되면 증가 될 수밖에 없다. 특히 센서에서 보내야 하는 패킷의 증가는 센서 노드의 연산량을 증가 시키고 추가적인 전력을 소비하게 되기 때문에 유비쿼터스 센서 네트워크와 같은 제한적인 환경에서는 중요한 요인이 된다. 먼저 본 논문의 구현 환경인 CC2420 RF 칩에서 사용하는 패킷의 구조는 (그림 10)과 같다[16].



(그림 10) IEEE 802.15.4 프레임 구조

(그림 10)에서 PHY(Physical Layer)에서 사용하는 헤더 부분은 6 byte이고 MAC(Medium Access Control) Layer에서 사용하는 헤더 부분은 5 byte에서 25 byte 이다. 그리고 (그림 10)에서 Frame Payload에 해당되는 부분은 센서 운영체제나 어플리케이션 레벨에서 사용하는 부분이다. 본 논문의 구현 환경인 Qplus에서 사용하는 패킷의 크기, 즉 Frame Payload의 부분은 7byte ~ n byte 이다. 그 중 센서에서 하나의 일반적인 데이터 값을 보내게 되는 기본 패킷의 크기가 7 byte 이기 때문에 전송되는 패킷의 크기는 11+(0+20)+7 byte가 된다. 즉 최소 18 byte에서 최대 38 byte가 될 수 있다. 본 논문에서 제안하는 대칭키 기반의 더블키를 이용한 보안 정책의 패킷의 구조는 (그림 5)와 (그림 7)에서 설명하였다. 대칭키 기반의 더블키를 이용한 보안 정책의 경우 센서 노드에서 전송 되는 패킷 데이터 크기는 보안 관련 데이터 3 byte 포함하여 최소 10 byte이며 인증, 키 교체 시 패킷의 크기는 17 byte를 사용한다. 센서 노드에서 패킷 수신 데이터의 크기는 8byte 와 24 byte로 주로 인증, 키 교체 및 보안 레벨 조정 등에서 사용 한다. 그 중 인증, 키 교체의 패킷을 제외하고 센서 노드에서 센싱한 데이터를 기본 패킷으로 보안 정책이 포함된 후 패킷의 증가량을 보면 <표 3>과 같다.

<표 2>에서 보면 최소 크기의 데이터 패킷일 경우 16.7%의 증가율을 가져왔으며 패킷의 크기가 최대 일 경우 7.9%의 증가율을 보였다. TinyOS를 기반으로 연구 중인 TinySec, SNEP, MiniSec의 경우 증가율과 본 논문의 센서 네트워크 보안 프로토콜(uSNSP)과 비교하면 다음 <표 4>와 같다[17].

<표 4>에서 비교는 최대 패킷의 크기로만 이용하여 비교하였다. <표 4>에서 알 수 있듯이 본 논문에서 제안한 유비쿼터스 센서 네트워크에서 더블키를 이용한 경량 보안 프로토콜은 다른 보안 정책들에 비하여 센서 노드의 운영체제에 적은 연산 처리 증가율을 가져온다. 본 논문에서 구현을 Qplus에서 했지만 제안하는 보안 프로토콜은 센서 노드의 운영체제와 무관하게 실행 할 수 있기 때문에, TinyOS 위에서 동작을 할 경우 패킷의 증가율은 <표 4>에서 MiniSec와 같은 8.3 %가 될 것 이다. 본 논문의 시뮬레이션은 CC2420 RF 칩에서 하였다. CC2420의 경우 전송 시 128 byte를 전송 할 수 있으며 그 전력 소비량은 17.5 mA 가 소비 된다[18]. 위에서 살펴 본 패킷양의 증가는 실제로 센서 노드에서 데이터를 전송 할 때 128 byte 까지는 전송 소비 전력은 증가 되지 않는다. 따라서 <표 3>의 증가율은 센서 노드에서 패킷 제작을 위한 CPU 연산에만 전력이 추가 적으로 소비가 될 뿐 전송 시에는 같다고 볼 수 있다.

5.2 제안한 프로토콜 실행 시 추가적인 패킷 전송량 증가

다른 보안 정책과 본 논문에서 제안하는 정책의 차이점은 다른 보안 정책은 센서 노드에서 키를 생성하고 관리하게 된다. 혹은 기본 인증 서버를 두고 그 서버를 통하여 인증을 하는 방식을 이용한다. 따라서 키 생성, 관리, 인증 관련

〈표 3〉 기본 패킷에 대한 보안 정책 후 패킷 증가 비교 (단위 byte)

	헤더 크기	보안 크기	데이터 패킷	최소 크기	최대 크기	증가율 (%)
기본 패킷	11+(0+20)	-	7	18	38	-
더블키 정책 포함 시	11+(0+20)	3	10	21	41	16.7, 7.9

〈표 4〉 다른 보안 정책과 패킷 증가율 비교 (단위 byte)

	패킷 크기	보안 패킷	총 패킷	증가율 (%)
TinyOS	36	-	36	-
TinySec	36	5	41	13.9
SNEP	36	8	44	22.2
MiniSec	36	3	39	8.3
Qplus	38	-	38	-
uSNSP	38	3	41	7.9

〈표 5〉 인증, 키 교환 보안 정책에 사용하는 추가 패킷 크기 (단위 byte)

		헤더 크기	기본 패킷	최소 크기	최대 크기	증가율 (%)
기본 패킷		11+(0+20)	7	18	38	-
더블키 정책 포함 시	전송	11+(0+20)	10,10,17	21	48	16.7, 26.3
	수신	11+(0+20)	10,17	21	48	16.7, 26.3

모듈이 센서 노드에 포함이 된다. 유비쿼터스 센서 노드의 하드웨어의 제약을 생각한다면 센서 노드의 연산량 증가가 필수적인 이러한 방식들은 부하가 커질 수밖에 없고 전력도 낭비하게 된다. 본 논문에서 제안하는 유비쿼터스 센서 네트워크에서 더블키를 이용한 경량 보안 프로토콜에서는 센서 노드의 연산량 줄여서 전력 소비를 줄이는 것이 목적이다. 그렇기 때문에 본 논문에서 사용하는 보안 정책은 인증, 키 교환 프로토콜에서는 센서 노드의 연산량을 없애는 대신 안전한 방식의 인증 관련 패킷 전송으로 대처를 하였다. 센서 노드에서 인증 관련, 키 관리 및 생성 관련 모듈이 포함되었을 경우와 보안 패킷 전송 방식과 직접적인 비교는 불가능 하지만 본 5.2절의 성능 평가에서는 추가적인 패킷 전송으로 인한 전력 증가에 대해서만 평가를 한다.

본 논문에서 인증, 키 전송 관련해서 센서 노드와 서버 간의 추가적인 패킷 증가는 <표 5>와 같다. 송수신 칩의 전송과 수신시 소비되는 전력이 차이가 나기 때문에 <표 5>와 같이 전송과 수신으로 나누어서 계산을 하였다. 본 논문에서 제안하는 보안 정책에서 인증과 키 교환 시 패킷의 크기가 128 byte를 넘지 않기 때문에 한 번 전송 또는 수신되는 패킷에 소비되는 전력은 일정하다고 볼 수 있다.

본 논문에서 제안하는 유비쿼터스 센서 네트워크를 위한 경량 보안 프로토콜에서 인증(a) 및 키 교체(k) 프로토콜을 사용하여 소비되는 총 전력(P)은 사용 횟수를 t, 전송 횟수를 st, 전송 소비 전력을 sc, 수신 횟수를 rt, 수신 소비 전력을 rc 라고 한다면 다음 식 (1)과 같이 표현 할 수 있다.

$$P = (At \times (Ast \times sc + Art \times rc)) + (Kt \times (Kst \times sc + Krt \times rc)) \quad (1)$$

대칭키 기반의 더블키를 이용한 인증 프로토콜을 사용 할 때 센서 노드에서 송신 및 수신 되는 횟수는 (그림 2)에서처럼 송신 3회, 수신 3회로 이루어진다. 또한 키 교체 프로토콜을 사용 시에는 (그림 3)에서처럼 송신 2회 수신 3회로 이루어진다. 그리고 본 논문의 구현 하드웨어 사양에서 송수신을 담당하는 칩의 경우 전송 시 17.4mA 그리고 수신 시 18.8mA를 소비한다. 센서 노드는 인증 프로토콜을 한번만 하게 되며, 키 교체 프로토콜을 3개월마다 1번씩 사용한다고 하였을 경우 대칭키 기반의 더블키 인증, 키 교체 프로토콜을 적용한 센서 노드에서 1년에 소비되는 추가적인 전력은 473.4 mA가 된다.

5.3 센서 노드에 포함되는 모듈 크기 비교

만일 기존의 보안 정책을 센서 노드에 포함을 한다면 센서 노드에 포함이 되는 모듈은 키 생성 모듈, 키 관리 모듈, 암호 및 복호화 모듈, 키 교환 모듈이 반드시 포함이 되어 할 것이다. 하지만 제안하는 보안 정책의 대칭키 기반의 더블키를 이용한 인증, 키 교환 프로토콜은 암호 및 복호화 모듈, 초기 인증 모듈, 키 교환 모듈만 포함 하면 가능하다. 대칭키 기반의 보안 모듈을 모두 포함 시켜야 할 경우 컴파일 하기 전의 코드의 크기는 34675 byte 이상 포함해야 한다. 하지만 본 논문에서 키 생성 모듈을 제외하고 21194 byte가 포함이 되었다. 제안하는 보안 프로토콜 사용 시 약 38.9%의 코드를 제외하고 같은 보안 레벨의 효과를 얻을 수 있었다. 본 논문에서 제안하는 더블키 보안 프로토콜은 센서의 운영체제, 패킷의 라우팅 방식, 암호 및 복호화 알고리즘에 독립적인 프로토콜이기 때문에 경량화 된 대칭키 암호 알고리즘을 사용하는 어떠한 정책에서도 본 논문에서 제안한 경량 보안 프로토콜을 사용할 경우 38%의 코드 절감 효과를 가져 올수 있으며 센서 노드의 전력 및 처리량을 줄일 수 있을 것으로 기대된다.

6. 결론 및 향후 연구

유비쿼터스 컴퓨팅 발전에 가장 중요한 역할을 하는 무선 센서 네트워크는 기존의 보안 정책을 적용하기가 힘든 분야이다. 또한 유비쿼터스 센서 네트워크의 보안은 무선을 이용하기 때문에 생기는 무선 보안 문제와 센서 디바이스의 제한된 환경에 필요한 보안 정책을 수립해야 하는 문제점을 동시에 가지고 있다. 그 중 본 논문은 무선 센서 네트워크의 두 가지 보안 문제 가운데 센서 디바이스의 제한된 환경으로 생기는 보안의 문제점에 대해서 논의하고 그 해결 방법을 제시하였다.

유비쿼터스 센서 노드들은 제한된 전력을 이용하고 하드웨어적으로 작은 크기를 유지해야 하기 때문에 연산 처리에도 제약이 따르게 된다. 따라서 유비쿼터스 센서 노드들의 제한된 전력과 하드웨어를 고려한 인증 프로토콜, 암호화 알고리즘과 같은 보안 정책 등이 필요하다. 최근 이러한 제약 조건을 만족시키기 위해서 여러 보안 정책들이 연구 중에 있다. 하지만 현재 연구가 진행 중인 보안 프로토콜들은 특정 센서 운영체제에서 동작을 하거나 특정 플랫폼, 지정된 라우팅 방식 등을 사용해야 가능한 보안 정책들이었다.

본 논문은 유비쿼터스 센서 네트워크에서 센서의 운영체제와 플랫폼, 데이터 전송 라우팅 방식, 암호화 알고리즘에 종속적이지 않은 유비쿼터스 센서 네트워크에서 더블키를 이용한 경량 보안 프로토콜을 제안하였다. 제안한 프로토콜

은 스마트 오피스 및 스마트 홈과 같은 실제 환경에 맞도록 유비쿼터스 센서 네트워크에서 각 센서 노드들의 보안 알고리즘과 이를 관리 하는 서버 간의 인증 및 보안 정책에 중점을 두었다. 성능 평가 결과 다른 센서 네트워크 보안 정책에 비하여 적거나 같은 크기의 보안 패킷 증가만으로도 구현을 할 수 있었으며 특히 운영체제에 상관없이 적용 할 수 있어서 더 효과적이었다. 또한 인증과 키 교환의 작업을 센서 노드에서 하지 않게 함으로써 센서 노드의 부하를 최대한 줄였다. 인증 및 키 교환 프로토콜을 포함함으로써 인증과 키 관련 모듈을 제외 하였으며 단지 서버에서 생성된 패킷으로 인증과 키 교체 관련 통신을 통하여 인증과 키 교환 작업을 하도록 하였다. 추가적으로 들어가는 전력의 소비도 센서 노드에서 인증 및 키 생성, 교환, 관리 알고리즘이 포함 되는 것 보다 적은 전력이 소비되는 것으로 판단되었다. 제안한 유비쿼터스 센서 네트워크에서 더블키를 이용한 경량 보안 프로토콜은 스마트 오피스 및 스마트 홈과 같은 실제 환경의 유비쿼터스 센서 네트워크에 적합한 보안 프로토콜이라고 할 수 있다.

향후 연구 과제로는 센서로부터 전송 되는 데이터 값을 통하여 센서 노드 및 센서 네트워크를 감시 할 수 있는 알고리즘 개발과 이것을 토대로 센서의 상태를 보안 레벨에 맞게 조절 할 수 있는 시스템을 개발하는 것도 필요하다. 또한 센서 노드의 하드웨어 성능이 향상 되는 것을 예상하고 센서 노드에 다른 암호화 복호화 알고리즘을 포함하여 여러 가지의 암호화 작업을 선택 또는 결정해서 전송 할 수 있도록 하는 보안 알고리즘을 개발도 필요하며, 암호화의 키 크기를 달리 해서 네트워크와 센서 노드의 상태에 따라서 다른 키 크기의 암호화, 복호화 작업을 하여 저전력을 유지 할 수 있도록 하는 시스템을 개발하는 것도 앞으로의 주요 과제이다.

참 고 문 헌

[1] 서운석,신순자,구자동,임진수,"유비쿼터스 컴퓨팅 환경에서 보안 및 인증 서비스 방향 연구", 한국전산원, http://www.ipc.go.kr/servlet/download?pt=/ipckor/policy&fn=file.pdf

[2] 김신효, 강유성, 정병호, 정교일, "u-센서 네트워크 보안 기술 동향", http://ettrends.etri.re.kr/PDFData/20-1_093_099.pdf

[3] TinySec home page, http://www.cs.berkeley.edu/~nks/tinysec/

[4] Tieyan Li, "Security Map of Sensor Network", http://www.i2r.a-star.edu.sg/icsd/SecureSensor/papers/security-map.pdf, March.2005

[5] Zigbee Document 03322r6ZB, Security Services Specification Release0.80, April,2004.

[6] Adrian Perrig, Robert Szewczyk, Victor Wen, David Culler, J. D. Tygar, "SPINS:Security Protocols for Sensor Networks", Proceedings of Seventh Annal International Conference on Mobile Computing and Networks, July 2001.

[7] Adrian Perrig, Ran Canetti, J. D. Tygar, Dawn Song, "Efficient Authentication and Signing of Multicast Streams over Lossy Channels", IEEE Symposium on Security and Privacy, May 2000.

[8] 한종수,배성수,김경목, "유비쿼터스기술(RFID와 홈 네트워킹)", 도서출판세화, 2005.

[9] 손승원, "네트워크 보안 기술의 현재와 미래", 한국 정보 진흥원, 2004

[10] D.J.Malan, M.Welsh, and M.D. Smith. "A Public-Key Infrastructure for Key Distribution in TinyOS Based on Elliptic Curve Cryptography", Fist IEEE International Conference on Sensor and Ad Hoc Communication and Networks, 2004.

[11] B. C. Neumann and T. Ts'o, "Kerberos: An authentication service for computer networks", IEEE Commun. Mag., vol. 32, no. 9, pp. 33-38, Sep. 1994.

[12] Network Working Group X.509, http://www.ietf.org/rfc/rfc2459.txt

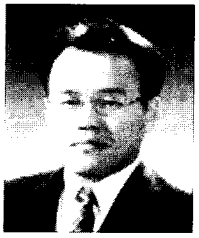
[13] TinyOS home page, http://www.tinyos.net/

[14] 손상철,윤미연,이광겸,신용태, "실시간 유비쿼터스 센서 네트워크에서 비밀 분산 기법에 기반한 효율적인 키 관리에 관한 연구", 한국 컴퓨터 종합 학술 대회 논문집 Vol.32, No.1(A) pp. 202~204, 2005

[15] IEEE 802.15 Task Group 4 (TG4), http://www.ieee802.org/15/pub/TG4.html

[16] Mark Luk, Ghita Mezzour, Adrian Perrig, Virgil Gligor, "MiniSec: A Secure Sensor Network Communication Architecture", http://www.truststc.org/pubs/197/ paper.pdf

[17] CC2420 Data, http://www.cs.ucsb.edu/~nchohan/docs/CC2420DataSheet.pdf



정 연 일

e-mail : zhungs@oslab.khu.ac.kr
2000년 경희대학교
컴퓨터공학과(공학석사)
2003년 경희대학교
컴퓨터공학과(박사수료)
2003년~현 재 경희대학교 컴퓨터공학과
박사과정
관심분야 : 유비쿼터스 컴퓨팅, 센서 네트워크 보안

이 승 룡

e-mail : sylee@oslab.khu.ac.kr
1978년 고려대학교 재료공학과 졸업
(학사)
1986년 Illinois Institute of Technology
전산학과(이학석사)
1991년 Illinois Institute of Technology
전산학과(이학박사)
1991년~1993년 Governors State University 조교수
1993년~현 재 경희대학교 컴퓨터공학과 교수
관심분야 : 유비쿼터스 컴퓨팅, 실시간 컴퓨팅, 유비쿼터스
미들웨어, 센서 OS, 시스템 보안