博士學位論文

# 유비쿼터스 센서 네트워크를 위한 경량 인증 프로토콜 기반의 보안 프레임워크

指導敎授 : 李 承 龍

慶熙大學校 大學院

電子計算工學科

鄭 然 一

2007年 8月 日

博士學位論文

# 유비쿼터스 센서 네트워크를 위한 경량 인증 프로토콜 기반의 보안 프레임워크

指導敎授 : 李 承 龍

慶熙大學校 大學院

電子計算工學科

鄭 然 一

2007年 8月 日

博士學位論文

# 유비쿼터스 센서 네트워크를 위한 경량 인증 프로토콜 기반의 보안 프레임워크

指導敎授 : 李 承 龍

이 論文을 工學博士 學位論文으로 提出함


慶熙大學校 大學院

電子計算工學科

鄭 然 一


2007年 8月 日

鄭然一의 工學博士 學位論文을 認准함

주심교수 (인)

부심교수 (인)

부심교수 (인)

부심교수 (인)

부심교수 (인)

# 慶熙大學校 大學院

2007年 8月 日

國 文 要 約

# 유비쿼터스 센서 네트워크를 위한 경량 인증 프로토콜 기반의 보안 프레임워크

# Light Weight Authentication Protocol based Security Framework for Ubiquitous Sensor Network

慶熙大學校大學院
電子計算工學科
鄭 然 一

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 이루어 질 수 있기 때문에 적합한 보안 정책이 필요하다. 특히 유비쿼터스 센서 네트워크의 센서 디바이스들은 제한된 전력을 이용하고 하드웨어적으로 작은 크기를 유지해야 하기 때문에 보안 정책 수립에 많은 제한이 발생하게 된다. 이러한 점을 해결하기 위해서 유비쿼터스 센서 네트워크 보안에 대한 많은 연구가 진행 중이지만 아직 해결해야 하는 문제점들을 가지고 있다. 더욱이 현재 연구가 진행 중인 보안 프로

토콜들은 특정 센서 운영체제에서 동작을 하거나 특정 플랫폼, 지정된 라우팅 방식 등을 사용해야만 그 보안 효과를 볼 수 있기 때문에 센서 노드의 운영체제나 플랫폼, 메시지 전송 라우팅 방식과 무관한 보안 정책이 필요한 실정이다.

본 논문에서는 유비쿼터스 센서 네트워크에서 센서 노드의 운영체제와 플랫폼 그리고 메시지 전송 라우팅 방식에 종속적이지 않은 경량 보안 프레임워크를 제안한다. 제안하는 경량 보안 프레임워크는 유비쿼터스 센서 네트워크의 센서 노드의 제한된 환경을 고려한 더블키 기반의 경량 보안 프로토콜과 공개키를 사용한 경량 보안 프로토콜을 제공한다. 또한 시큐어 넘버, 시큐어 카운터를 이용한 보안 정책 등 여러 보조 정책을 이용할 수 있도록 지원한다. 이런 정책들로 보안 관리자가 현재의 네트워크 상태 정보를 모니터링 하여 보안의 적정한 레벨을 선택하며 네트워크와 센서 노드를 관리할 수 있도록 지원하는 유비쿼터스 센서 네트워크를 위한 경량 보안 프레임워크를 제안한다.

성능 평가 결과 제안한 유비쿼터스 센서 네트워크를 위한 경량 보안 프레임워크는 기존의 센서 네트워크 보안 정책 보다 상대적으로 저전력을 사용하였으며 센서 노드에 포함 되는 보안 모듈의 경량화로 제안적인 센서 노드 환경에 적합한 보안 정책을 활용 할 수 있었다. 제안된 경량 보안 프레임워크는 스마트 오피스 및 스마트 홈과 같은 실제 환경의 유비쿼터스 센서 네트워크에 적합한 성능을 나타내었다.

# <목 차>

# <그림 목차>

# <표 목차>

# 제 1 장 서 론

## 1.1 연구 배경

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 무선 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 디바이스의 제한된 환경에 맞는 새로운 보안 정책을 수립해야 하는 문제점을 동시에 가지고 있다. 본 논문은 유비쿼터스 센서 네트워크의 두 가지 보안 문제 가운데 센서 디바이스의 제한된 환경으로 생기는 보안의 문제점에 대해서 논의하고 그 해결 방법을 제시하고자 한다.

유비쿼터스 센서 네트워크의 센서 디바이스들은 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 디바이스들의 제한된 전력과 하드웨어를 고려한 인증 프로토콜, 암호화 알고리즘과 같은 보안 정책들이 필요하다[61][73][65]. 최근 이러한 제약 조건을 만족시키기 위해서 여러 보안 정책들이 연구 중에 있다. 하지만 이러한 보안 정책에는 각기 해결해야 하는 몇 가지 문제점들이 있다. 대칭키를 이용한 보안 정책

을 사용 경우에는 암호화·복호화에 효율성을 보일 수 있지만 대칭키를 안전하게 공유해야 하는 문제점을 해결해야 한다. 대칭키를 이용한 보안 정책의 경우에는 특히 하나의 노드가 전복 되었을 경우에도 그 피해는 유비쿼터스 센서 네트워크 전체에 발생하게 된다. 브로드캐스트 메시지 인증의 경우 복잡한 절차가 요구되며 많은 트래픽을 발생시키기 때문에 많은 전력을 소모하게 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 센터나 계층적인 사전 키 분배 방식을 이용하는 연구[59][62]도 진행 중이지만 이러한 방식 역시 새로운 트래픽을 발생시키며 노드 자체의 연산을 증가시키기 때문에 실제로 구현을 하여 활용하기에는 적당하다고 볼 수 없다. 현재 센서 네트워크 보안에서는 대부분 상대적으로 모듈의 크기가 작고 암호화 처리속도가 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[69]. 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.

더욱이 현재 연구가 진행 중인 보안 프로토콜들은 특정 센서 운영체제에서 동작을 하거나 특정 플랫폼, 지정된 라우팅 방식 등을 사용해야만 그 보안 효과를 볼 수 있도록 연구가 되기 때문에 센서의 운영체제나 센서의 플랫폼, 라우팅 방식과 무관한 보안 정책이 필요한 실정이다.

## 1.2 연구 목적 및 방법

본 논문에서는 유비쿼터스 센서 네트워크에서 센서의 운영체제와 플랫폼, 데이터 전송 라우팅 방식, 암호화 방식에 종속적이지 않은 경량 보안 프레임워크를 제안한다. 제안하는 경량 보안 프레임워크는 스마트 오피스 및 스마트 홈과 같은 실제 환경의 유비쿼터스 센서 네트워크에서 각 센서 노드들의 보안 알고리즘과 이를 관리하는 서버 간의 인증 및 보안 정책에 중점을 두고 설계를 하였다.

본 논문에서 제안하는 경량 보안 프레임워크는 유비쿼터스 센서 네트워크의 제한된 센서 노드에 적합한 보안 알고리즘을 이용하며 센서 노드별로 다른 키를 관리하도록 하였다. 제안하는 보안 프로토콜은 유비쿼터스 센서 네트워크의 센서 노드의 제한된 환경을 고려한 대칭키 기반의 더블키를 이용한 경량 보안 프로토콜과 공개키를 사용한 보안 프로토콜로 구분할 수 있다. 또한 시큐어 카운터를 이용한 보안과 전송 거리 조절을 이용한 보안 정책을 이용하여 공격자의 공격에 능동적으로 대처를 할 수 있게 지원하며 유비쿼터스 센서 네트워크 상태를 모니터링 하여 현 센서 네트워크 상태에 맞는 보안의 적정한 레벨을 선택하고 관리할 수 있도록 지원한다.

본 논문에서 제안하는 더블키를 이용한 경량 보안 프로토콜 보안 정책의 특징은 다음과 같다. 첫째, 각 개체 간 인증을 할 경우에 기존의 인증 방식을 이용하여 구현 할 경우는 제한적인 하드웨어로 구성되어 있는 센서 노드들은 인증을 위한 알고리즘을 포함하고 있어야 한다. 이러한 방법의 접근은 센서 노드가 수행하는 연산량 증가를 가져오며 이러한 문제를 해결하기 위해서는 센서 노드의 하드웨어적인 증가가 필요 하게 된다[57]. 본 논문의 더블키를 이용한 보안 프로토콜은 경량 인증 프로토콜을 제안하여 센서 노드에는 상대적으로 크기가 작은 모듈이 포함되기 때문에 하드웨어의 증가 없이 사용할 수 있다. 또한 센서 노드에는 인증을 위한 최소한의 연산만을 증가시켜 안전한 인증을 할 수 있도록 하였다. 둘째, 기존의 공개키를 이용한 보안 정책과 대칭키를 이용한 보안 정책은 유비쿼터스 센서 네트워크에 적용

하기에는 다음과 같은 문제점을 각각 포함하고 있다. 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 이용하여 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 저전력으로 운영 할 수가 있다.

일반적으로 센서 네트워크에서는 센서 노드의 제한적인 하드웨어 환경 때문에 암호키의 저장 및 관리에 어려움이 많은 공개키 방식의 암호 정책 보다 대칭키 방식의 보안 정책이 대부분 제안되고 있다. 하지만 키 관리 및 경량화 된 암호화 알고리즘에 대한 문제점만 해결이 된다면 대칭키 방식의 암호 정책 보다는 공개키 방식의 암호 정책이 유비쿼터스 센서 네트워크에서 더 많은 장점을 가지고 있다. 본 논문에서 제안하는 공개키 보안 정책의 방식은 센서 네트워크의 각 노드를 평등한 관계로 보는 것이 아니라 센서 노드와 데이터 수집 서버와의 관계로만 인식을 하고 기존 공개키 보안 정책을 센서 네트워크에 맞도록 수정하여 센서 노드의 하드웨어 능력에 따라서 인증 및 암호화 알고리즘을 사용할 수 있도록 하는 공개키 기반의

보안 정책을 제안한다. 본 논문에서 제안하는 공개키를 사용한 경량 보안 프로토콜의 특징은 다음과 같다. 첫째, 기존의 공개키 보안 정책을 사용 할 경우 인증 혹은 암호화를 원하는 양쪽에서 키 관리를 해야 하며 이러한 진행은 다대다의 인증을 해야 할 경우 관리해야 할 공개키의 수가 많아져서 센서 네트워크의 제한적인 하드웨어 환경에는 적합하지 않기 때문에 실제로 센서 네트워크에서는 구현을 하지 못하였다. 하지만 제안하는 공개키 기반의 보안 프로토콜은 센서 노드에서는 공개키 관리가 필요 없고 공개키로 암호화 작업만 필요로 하기 때문에 센서 노드에서 다량의 키를 관리 할 필요가 없으며 또 여러 개의 공개키를 보관 하거나 인증서버에 패킷을 보내는 작업을 할 필요도 없게 된다. 따라서 공개키를 이용한 보안 정책을 사용하지만 실제 센서 노드의 공개키 관리를 위한 연산량은 줄어드는 장점이 있다. 둘째, 대칭키 보안 정책은 데이터의 암호화와 복호화의 작업을 해야 하기 때문에 센서 노드에서 보안 정책이 없을 경우 보다 많은 연산 처리해야 한다. 더욱이 센서의 종류에 따라서 암호화, 복호화 연산을 모두 할 경우 소비되는 전력은 증가 하게 된다. 제안하는 공개키 기반의 보안 프로토콜은 이러한 경우에 센서 노드에서 복호화 작업을 없애고 센서 노드에서는 암호화 작업만 가능하게 해서 데이터를 전송하는 방식을 사용한다. 따라서 어떠한 공개키 보안 알고리즘과 상관없이 상대적으로 적은 연산량만으로 인증 및 암호화 작업이 가능할 수 있게 한다. 셋째, 제안하는 공개키 기반의 인증 프로토콜은 센서 노드에 인증 및 공개키 관리의 부담이 줄어들고 복호화 작업을 제외시킴으로 저전력의 효과를 볼 수 있다. 이러한 점은 제한된 하드웨어에서도 보안 정책을 구현 할 수 있기 때문에 센서 노드의 부담을 최대한 줄이고 상대적으로 안전한 보안 정책을 펼칠 수 있게 한다.

제안하는 유비쿼터스 센서 네트워크 경량 보안 프레임워크에는 보안 프레임워크 구성을 위한 보조적인 보안 정책으로 시큐어 카운터 정책과 전송 제한 정책을 이용할 수 있게 하며 다른 보안 정책과 통합적으로 보안 레벨을 조절 할 수 있도록 하는 장점이 있다. 시큐어 카운터와 전송 제한 정책은 유비쿼터스 센서 네트워크에서 보안을 수동적으로만 하는 것이 아니라 능동적으로도 사용할 수 있게 할 수 있으며

사용자 및 관리자의 판단에 의해서 여러 레벨에 다른 보안 정책을 센서 노드에 적용하여 안전한 센서 네트워크를 유지 할 수 있도록 한다. 또한 현재의 유비쿼터스 센서 네트워크의 상태를 계속 해서 모니터링 하여 현 네트워크의 상태에 맞는 보안 정책을 결정하여 네트워크의 안정성을 유지 시킬 수 있다.

## 1.3 연구 결과

제안한 유비쿼터스 센서 네트워크를 위한 경량 보안 프레임워크는 스마트 오피스 및 스마트 홈과 같은 실제 환경의 유비쿼터스 센서 네트워크에 적합하다고 할 수 있다. 실제 센서 노드의 제한된 하드웨어의 특징에 맞게 선택된 알고리즘을 이용하여 보안 정책을 관리 할 수 있도록 해주며 보안 정책이 포함이 되었음에도 상대적으로 저전력으로 보안 정책을 활용 할 수 있어서 효과 적이다.

성능 평가 결과 다른 센서 네트워크 보안 정책에 비하여 적거나 같은 크기의 보안 패킷 증가만으로도 구현을 할 수 있었다. 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 오직 서버에서 생성된 패킷으로 인증과 키 교체 관련 통신을 통하여 인증과 키 교환 작업을 하도록 하였다. 이러한 작업을 통하여 추가적으로 들어가는 전력의 소비를 줄였다. 또한 모니터링을 통하여 현재의 센서 노드의 상태나 네트워크의 상태를 파악할 수 있도록 하여 안전한 네트워크 상태를 유지 할 수 있도록 하였으며 공격자의 공격에 능동적으로 대처 할 수 있도록 하여 방어적인 센서 노드를 구성 하였다.

본 논문은 제 1장의 서론에 이어 제 2장에서는 유비쿼터스 환경 및 유비쿼터스 센서 네트워크 환경과 유비쿼터스 센서 노드 하드웨어에 대하여 설명하며, 하드웨어 보안 플랫폼과 유비쿼터스 센서 네트워크의 보안 목적 및 범위 그리고 센서 네

트워크 보안 정책과 관련된 연구에 대해서 소개를 한다. 3장에서는 유비쿼터스 센서 네트워크를 위한 경량 보안 프레임워크에서 사용되는 두 가지 방식의 보안 프로토콜에 대해서 설명한다. 4장에서는 유비쿼터스 센서 네트워크의 상태 및 센서 노드의 보안 상태를 감시하는 감시 프로토콜에 대해서 설명한다. 5장에서는 3장과 4장의 프로토콜을 이용하여 경량화 된 보안 프레임워크에 대한 구조 및 구현에 대해서 기술하고 6장에서 경량 보안 프레임워크에 대하여 기존 시스템과의 비교 분석을 통해 성능 평가를 하며, 끝으로 7장에서는 결론 및 향후 연구 방향에 대해 기술한다.

# 제 2 장 관련 연구

네트워크 보안 장비, 트래픽 조절 장비 등의 일부 장비의 보안에 관한 정의만으로 네트워크의 보안을 정의하는 것은 논리적으로 맞지 않다. 데이터 보호, 장비 보호 등을 담당하는 시스템 레벨뿐만 아니라 트래픽 조절, 경로 보호, 네트워크 신뢰성 보장 기술, 통합 보안 관리 기술 등이 포함된 종합적인 기능으로서 이들 시스템들을 유기적으로 엮는 개념으로서 정의하는 것이 맞다[9]. 마찬가지로 유비쿼터스 센서 네트워크 환경에서 요구되는 보안 서비스를 만족하기 위해서 먼저 유비쿼터스 센서 네트워크의 일반적인 환경의 구조와 유비쿼터스 센서 네트워크 보안의 목적 그리고 요구되는 보안의 범위에 대하여 명확한 정의가 필요하다. 또한 유비쿼터스 센서 네트워크를 보호하기 위한 보안 프로토콜, 인증, 데이터 보안, 네트워크 감시 등의 전반적인 보안 기술과 네트워크 전체의 보안 프레임워크 정립이 필요하다.

본 2장에서는 1절에서 유비쿼터스 센서 네트워크 환경에 대하여 정의하고, 2절에서는 유비쿼터스 센서 네트워크의 구조와 실제 유비쿼터스 센서 네트워크에서 요구되는 보안의 내용 및 범위에 대하여 설명한다. 3절에서는 현재 유비쿼터스 센서 노드 플랫폼에 대해서 기술한다. 또한 4절에서는 하드웨어 보안 플랫폼에 대하여 설명하고 5절에서는 기존의 센서 네트워크 보안 정책에 대해서 기술한다.

## 2.1 유비쿼터스 센서 네트워크 환경

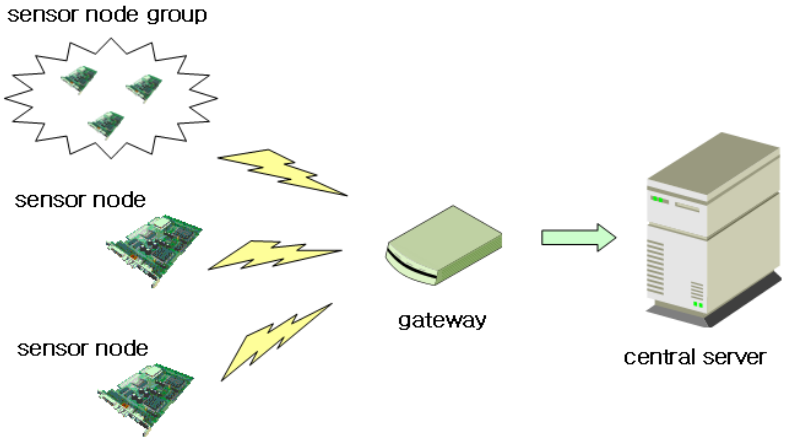
유비쿼터스(Ubiquitous)란 라틴어에서 유래한 것으로 언제, 어디서나 존재한다는 뜻으로, 유비쿼터스 컴퓨팅이란 통신 기능을 지닌 센서 또는 초소형 컴퓨터를 사물 또는 주변 모든 공간에 내장시켜 사람이 보이지 않는 네트워크로 연결시킨 후 실세계의 상황을 인식 하여 인간 누구에게나, 언제, 어디서나, 어떠한 서비스를 어떠한 기기로도 사람들에게 눈에 보이지 않게 정보를 제공하고 활용할 수 있는 인간중심

의 새로운 컴퓨팅 모델을 말한다. 이를 위하여, 미국, 유럽, 일본에서는 유비쿼터스 사회를 구축하기 위한 연구개발이 활발히 진행 중이다. 이들 과제들은 공통적으로 자율형의 센서와 통신 플랫폼을 개발하며, 이동성 보장을 위한 연구를 수행하고 있다. 상황을 인지하기 위한 자율형 센서는 저가·저전력 초소형 칩으로 사물 또는 공간에 장착되어 다양한 정보를 수집하고 통신을 하는 기능을 수행한다[70]. 그리고 통신 플랫폼은 다양한 센서들 간에 동적으로 변화하는 Ad-hoc 센서 네트워크를 구성하고 이를 기존의 유무선 네트워크와 연동시키는 역할을 하며, 컴퓨팅 디바이스 및 센서의 초소형화로 휴대 및 부착을 용이하게 함으로써 이동성을 보장한다. 이처럼 유비쿼터스 컴퓨팅 환경을 구축하기 위한 핵심 기술로는 첫째, 모든 사물 및 디바이스, 센서 등에 내장되기 위한 저전력, 초소형 칩 기술, 둘째, 다양한 부가 디바이스, 동적으로 변화하는 네트워크를 구성하기 위한 Ad-hoc 네트워크 및 센서 네트워크 기술, 셋째, 시스템 통합 및 다양한 이기종 시스템을 통합할 수 있는 개방형 소프트웨어 표준 아키텍처 및 미들웨어 기술 등이 필요하다. 또한, 이들 칩과 디바이스, 네트워크 기술을 응용하여 스마트 홈, 자동화된 공장, 건물 등의 다양한 분야에 적용될 수 있는 유비쿼터스 특성에 맞는 서비스 솔루션도 요구된다[77].

본 논문에서 기술하는 유비쿼터스 센서 네트워크는 정의 방식에 따라 약간의 차이가 있을 수 있으나 일반적으로, 제한된 전력으로 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. [그림 2-1]과 같이 유비쿼터스 센서 네트워크의 환경은 센서 노드 혹은 센서 노드 그룹의 데이터가 게이트웨이나 혹은 AP(Access Point)로 데이터로 모아지고, 모아진 데이터는 중앙 관리 서버에서 데이터의 분석 및 관리, 저장을 하게 된다.

센서 네트워크와 Ad-hoc 네트워크는 유사하지만 여러 가지 차이점이 있다[74]. 본 논문의 유비쿼터스 센서 네트워크의 보안적인 측면에서 그 차이점을 기술 하자면 다음과 같다. Ad-hoc 네트워크의 경우 네트워크가 형성된 후 노드 객체의 네트워크내의 가입 및 탈퇴가 자유로이 할 수 있어야 하며 그렇기 때문에 노드의 이동

성을 고려해야 한다[60][72]. 하지만 센서 네트워크의 경우 각 센서 노드가 임의대로 네트워크내의 가입 및 탈퇴가 불가능하고 센서 자체의 이동성은 지원하지만 어플리케이션에 의존하지 않는다[14][64]. 또한 센서 노드는 Ad-hoc 네트워크의 노드들 보다 제한된 하드웨어로 구성이 되며 인증 되지 않은 센서의 침입, 위조 혹은 변조된 데이터를 전달하거나 라우팅 과정에서 정보가 유출되는 것을 막아야 한다. 이러한 관점에서 본 논문에서는 유비쿼터스 센서 네트워크 환경에서 서버와 센서 노드들에 대해 다음과 같이 재정의 할 수 있다[그림 2-2].

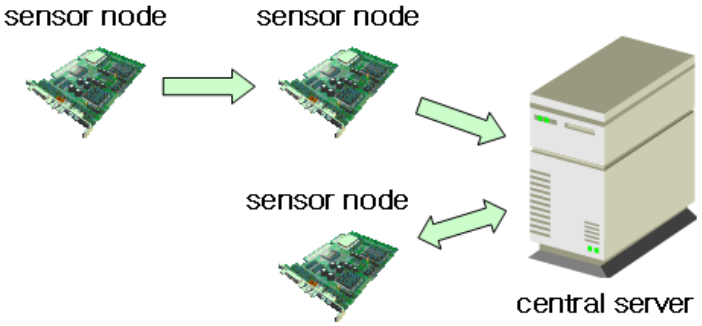



[그림 2-1] 유비쿼터스 센서 네트워크의 일반적인 환경

재정의 하는 유비쿼터스 센서 네트워크에서, 중앙 서버의 특징은 센서 노드로부터 수집된 데이터를 분석 및 저장을 담당하고 센서 노드의 수와 센서 네트워크의 구조에 대해서 파악 및 관리를 할 수 있어야 한다[75]. 또한 중앙 서버는 센서 노드보다 하드웨어적으로 제약점이 적어서 센서 네트워크 관리 및 데이터 처리에 필요한 하드웨어 환경을 충분히 이용할 수 있다.

그리고 본 논문에서 재정의 하는 유비쿼터스 센서 네트워크에서 센서 노드의 특징은 다음과 같다. 센서 노드는 브로드캐스트 통신 환경을 사용하며 중앙 서버와 양방향 통신 혹은 단방향 통신이 가능하다. 센서 노드는 각각의 고유한 인식자(ID)를 가지고 있으며 전원, 컴퓨팅 능력 등의 제한된 하드웨어로 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 통신은 불가하며 수집된 데이터를 라우팅 하는 연산만 가능하며 센서 노드 사이에는 전달 받은 데이터를 분석 하거나 재전송을 요구하지는 않는다.




[그림 2-2] 센서 노드와 중앙 관리 서버와의 관계

## 2.2 유비쿼터스 센서 네트워크 보안의 목적 및 범위

일반적으로 보안의 목적은 인증(authentication), 기밀성(confidentiality), 무결성(integrity), 부인봉쇄(non-repudiation)에 기준을 두고 있다. 그 중에 메시지를 전송했다는 사실을 부인 하지 못하는 부인봉쇄 항목은 센서 노드에서는 데이터 전송을 하고 중앙 관리 서버에서는 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[72]. 따라서 유비쿼터스 센서 네트워크에서 보안의 기준이 될 수 있는 네 가지 항목은 다음과 같다.

- 인증 (authentication) : 메시지가 송신 주체라고 주장하는 송신자로부터 왔다는 확신을 갖게 해주는 서비스.

- 기밀성 (confidentiality) : 공격으로부터 데이터를 보호하는 것을 말하는 것으로 데이터가 공격자에게 노출되지 않도록 하는 서비스.

- 무결성 (integrity) : 데이터가 중간에 복제되거나, 추가, 수정되거나, 재전송되지 않았다는 것을 보장해주는 서비스.

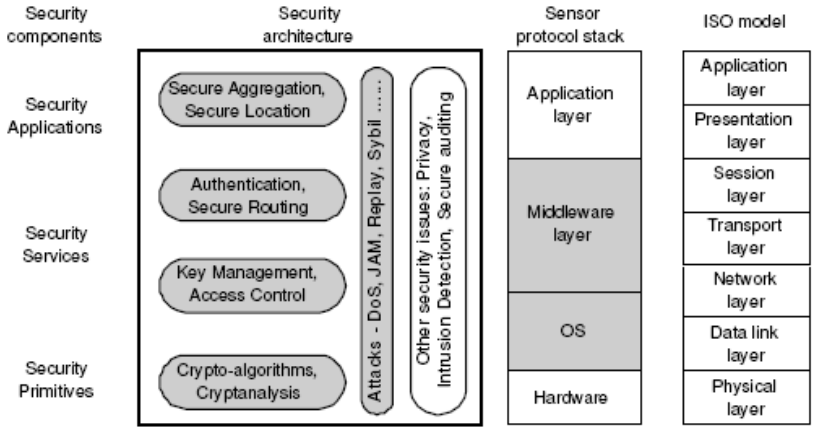
- 가용성 (availability) : 인가된 시스템이 자원에 접근할 필요가 있거나 사용하고자 할 때 시스템의 성능에 따라 시스템 자원에 접근할 수 있도록 하는 서비스

<표 2-1> 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<br>• 센서 노드 내에서 암호화에 사용하는 키 관리<br>• 센서 노드 환경에 맞는 경량화 된 암호화 알고리즘 |
| 무결성 | • 센서 노드에서 보내는 데이터의 무결성<br>• 공격자로부터 재전송 및 서비스 거부 공격에 대처 |
| 가용성 | • 센서 네트워크에 센서 노드의 가입 및 탈퇴 처리<br>• 센서 노드에서 중앙 관리 서버로의 안전한 데이터 라우팅<br>• 공격자로부터 재전송 및 서비스 거부 공격에 대처 |

유비쿼터스 센서 네트워크에서 센서 노드는 주변 환경의 데이터를 수집하여 중앙 처리 서버로 전송을 하며 중앙 처리 서버에서는 센서 노드들로부터 수집된 데이터

를 분석, 처리, 저장하는 구조를 가지고 있다. 이러한 센서 네트워크 구조에서 고려해야 할 보안의 사항 및 범위는 다음 <표 2-1>과 같이 정리 할 수 있다. <표 2-1>과 같은 보안 요구 사항을 만족하기 위해서 유비쿼터스 센서 네트워크에서 갖추어야 할 보안 구조로 [그림 2-3]과 같은 보안 구조를 제시하고 있다[4].




[그림 2-3] 센서 네트워크의 보안 구조도

본 논문에서 제안하는 유비쿼터스 센서 네트워크를 위한 경량 보안 프레임워크의 경우 제한이 많은 센서 노드를 위한 더블키를 이용한 인증 프로토콜과 공개키를 사용한 인증 프로토콜로 센서 노드와 중앙 관리 서버간의 인증을 담당하게 된다. 또한 데이터의 기밀성에 해당하는 암호화키에 대한 관리와 데이터의 무결성을 지원하기 위한 정책을 지원하며 공격자가 네트워크에 대하여 공격을 할 경우 능동적으로 대처를 할 수 있는 정책을 포함 하고 있다.

## 2.3 유비쿼터스 센서 노드 플랫폼

유비쿼터스 센서 네트워크란 인간과 컴퓨터간의 커뮤니케이션에 일상생활의 산재

된 사물과 물리적 대상을 추가시켜 협력 네트워크를 구성하는 것으로, 필요로 하는 모든 곳에 수많은 센서 노드들을 부착하여 자율적으로 정보를 수집, 관리 및 제어하는 시스템이다. 즉 물리 공간에 빛, 소리, 온도, 움직임 같은 물리적 데이터를 센서 노드에서 감지하고 측정하여 중앙의 기본 노드로 전달하는 구조를 가진 네트워크이다[42][76]. 센서 노드는 2005년 노드 당 가격이 10~100달러 선에서 시장이 안정화 되는 2007~2010년에는 노드 당 1~10달러 정도로 진입할 것으로 예상된다[43]. 또한 2010년에는 1억3천 ~1억8천 개의 센서 노드가 사용될 것으로 예상하고 있다[44][68].

지금까지 개발된 블루투스(Bluetooth)나 무선 랜(Wireless LAN)등의 무선 네트워크 기술들은 반드시 컴퓨터나 PDA 같은 고급 컴퓨팅 장치를 필요로 했다. 하지만 센서 네트워크의 특성상 센서 노드는 가격과 유지 보수 문제 등으로 고급 컴퓨팅 장비를 사용할 수 없다. 그래서 센서 노드를 위한 하드웨어 플랫폼은 다음과 같은 특징을 가져야 한다[42].

- 한번 배치되면 유지보수가 어렵기 때문에 강인한 구조와 저전력으로 동작되도록 설계되어야 한다.
- 응용 분야에 따라 크게 달라질 수 있기 때문에 어떠한 구조에도 효과적으로 사용될 수 있도록 유연성과 모듈성을 갖추고 있어야 한다.
- 유비쿼터스 컴퓨팅 환경에 맞게 무선 네트워크 기술을 지원해야 한다.
- 많은 센서 노드가 산재되기 위해 가격이 저렴하고 크기가 작아야 한다.

이와 같은 특징을 만족시키기 위해 많은 하드웨어와 노드들이 개발되고 있다[63]. 본 장의 2.3절에서는 센서 노드의 하드웨어 플랫폼 중 핵심 부품인 MCU, 무선 모듈, 그리고 현재 개발되어 사용 중인 센서 노드들에 대해서 설명한다[66].

- **MCU (Micro Controller Unit)**

스마트 센서 노드에 들어가는 저전력 CPU의 경우 Atmel[45], ARM[46],

Motolora 등에서 개발되고 있다. 기존 국내 MCU 시장을 보면 마이크로칩스사의 PIC 계열과 인텔의 8051이 공개된 회로와 다양한 어플리케이션 제공으로 많이 사용되었다. 하지만 센서 네트워크에서는 에너지 효율성을 위해 대부분 8bit 계열에 낮은 클록을 가지는 RISC(Reduced Instruction Set Computer) CPU가 많이 사용된다.

최근 들어 Atmel의 AVR ATmega 시리즈가 널리 이용되고 있는데 이는 PIC처럼 RISC 및 하버드 구조를 가지는 고성능 8bit 마이크로 컨트롤러이면서 프로그램용 코드 메모리로 플래시 메모리를 내장하고 있어 쉽게 반복적인 프로그래밍이 가능하기 때문이다. 또한 8bit RISC 계열의 MCU인 AVR은 한 클록 당 한 명령어를 처리하기 때문에 동일 클록에서 동작 시 PIC나 8051을 채택한 MCU보다 몇 배 이상 빠르다. 그래서 가장 늦게 출시되었음에도 불구하고 단시간 내에 많은 인기를 얻었으며 현재 USN에 가장 많이 사용되고 있다.

- **무선 통신 모듈 (Radio Frequency 모듈)**

센서 네트워크 노드를 위한 RF 모듈로는 IEEE 802.15.4/Zigbee가 사실상 표준으로 사용되고 있다[43]. Zigbee는 저가, 저전력을 특징으로 IEEE 802.15.4의 표준화된 PHY/MAC 레이어를 기반으로 하여 상위 프로토콜 및 어플리케이션을 규격화한 단거리 무선 통신에 적합한 스펙이다.

Zigbee가 센서 네트워크에 많이 쓰이는 이유 중 하나는 바로 저렴한 가격이다. 시스템에 Zigbee를 구현하는 비용이 현재 약 2달러에 불과하기 때문에 많은 센서 노드를 설치해야 하는 USN에 적합하다. 또 802.11x wireless LAN이나 IEEE 802.15.3 HR-WPAN, Bluetooth 보다도 구현 측면에서 훨씬 간단한 것도 Zigbee의 장점으로 들 수 있다. 실제 Zigbee 실행 코드 사이즈는 32KB 이하로 구현이 가능하며 RFD(Reduced Function Device)와 같은 단순한 기능을 가진 노드의 경우 4KB 이하의 사이즈로 구현이 가능하다[47]. 전력 소모를 극소화한 것도 가장 큰 장점으로 들 수 있는데 네트워크 안에 하나의 coordinator를 명하여 송/수신의 활동에 필요한 경우에만 동작하고 나머지 시간은 sleep 모드로 보내기 때문에 한 번 배터리를 장착하면 최대 2~3년 정도 사용이 가능하다[42].

주요 Zigbee 반도체 업체로는 국내의 경우 레이디오펄스[48]가 Zigbee 칩을 생산하고 있으며 외국의 경우 Chipcon, Motorola, Texas Instruments, ST Microelectronics, Ember 등이 있다.

- **Crossbow Mica Series**

미국 국방성의 DARPA 프로젝트의 스폰서를 받아 개발된 미국 버클리 대학 Mote 시리즈는 미국 정부와 관련 대하고가 기업의 노력으로 가장 널리 사용되는 하드웨어 플랫폼으로 자리 잡고 있다[49]. 이는 하드웨어뿐만 아니라 TinyOS라는 센서 네트워크용 OS와 각종 시뮬레이터 및 공개 애플리케이션을 통한 인프라가 구축되었기 때문이다. 1999년 처음으로 WeC라는 플랫폼이 개발되었으며 매년 Rene, dot, Mica, Mica2, MicaZ와 같은 센서 노드 하드웨어가 개발되어 왔다. 해당 노들들은 버클리 출신이 만든 벤처회사 Crossbow,를 통해 상업화되어 시장에 공급되고 있다. [그림 2-4]는 Mote 시리즈의 하드웨어 사양으로 그 동안 발표된 메인 보드별 스펙을 나타낸다. 이 메인 보드에 온도, 조도, 지자기 센서 등과 같은 센서를 탑재한 센서 보드를 스택 형식으로 장착할 수 있다. CPU 스펙의 경우 앞서 언급한 Atmel의 Atmega 시리즈를 사용하고 있으며 노드별 가장 큰 차이는 RF 인터페이스에서 찾아 볼 수 있는데 Mica는 916 MHz의 라디오 트랜시버를 사용한 데 비해 Mica2는 433/868/915 MHz의 다양한 무선 밴드를 지원한다. 또한 MicaZ 같은 경우 Mica2와 비슷하지만 RF 모듈을 Chipcon사의 CC2420을 사용하여 Zigbee를 지원한다.

- **Intel Mote (iMote)**

버클리대의 산업체 파트너인 인텔에서 개발된 Mote 시리즈로 ARM 기반의 마이크로프로세서를 사용한다. 이는 센서 네트워크의 응용분야를 조사하고 다양한 응용을 위해 성능을 향상시킨 디자인이다. 센싱 정보의 복잡한 계산이나 라우팅 등의 상위 레벨의 정보 처리, 보안 등에 더 치중하기 위해 Intel의 iMote는 강력한 32bit ARM7TDMI CPU를 사용했다. 12MHz의 클록으로 기존 Mica Mote 보다 4배 정도

의 성능을 보이며, 메모리 측면에서도 Mica 시리즈보다 훨씬 큰 512KB의 프로그램 메모리와 64KB의 RAM을 사용한다[50]. RF 모듈은 Zeevo사의 2.4 GHz band의 Bluetooth를 사용하여 최대 720kbps의 전송률을 가지기 때문에 250kbps의 전송률을 가지는 Zigbee 보다 훨씬 많은 데이터를 전송 할 수 있으며, 크기는 3X3 cm 정도로 아주 작다[51].

| Mote | WeC | Rene | Rene2 | dot | Mica | Mica2 | MicaZ |
|---|---|---|---|---|---|---|---|
| Released | 1998 | 1999 | 2000 | 2001 | 2002 | 2003 | 2004 |
| Clock | 4MHz | 4MHz | 4MHz | 4MHz | 4MHz | 7.37MHz | 7.37MHz |
| CPU | Atmel 90LS8535 | Atmel 90LS8535 | Atmel Atmega164 | Atmel Atmega163 | Atmel Atmega103L | Atmel Atmega128L | Atmel Atmega128L |
| Program Memory(KB) | 8 | 8 | 16 | 16 | 128 | 128 | 128 |
| RAM(KB) | 0.5 | 0.5 | 1 | 1 | 4 | 4 | 4 |
| RF Module | RFM TR1000 | FRM TR1000 | FRM TR1000 | FRM TR1000 | RFM TR1000 | Chipcon CC1000 | Chipcon CC2420 |
| Modulation | OOK | OOK | OOK | OOK | ASK | FSK | O-QPSK |
| Data Rate(KB) | 10 | 10 | 10 | 10 | 40 | 38.4 | 250 |
| TinyOS | - | V0.6.1 | V0.6.1 | V1.0 up | V1.0 up | V1.0 up | V1.1.7 up |

[그림 2-4] Crossbow사의 센서 노드 하드웨어 스펙

- **Moteiv Telos, Tmote SKY**

2004년 초에 출시된 Telos는 IEEE 802.15.4 표준을 지원하는 최초의 센서 노드이다. MCU로는 Texas Instruments의 MSP430을 사용하여 Chipcon사에서 개발된 CC2420 Zigbee 칩을 RF 모듈로 사용한다. Telos의 경우 Revison A, B가 있는데 두 제품의 차이는 RAM, flash가 다르고 그 외에는 거의 비슷하다. 일반적으로 Telos라고 하면 rev A를 지칭하며, rev B는 현재 생산이 중단되었다. 드리고 Telos rev B의 경우 TMote SKY라는 이름으로 변경되어 팔리고 있다[52]. Telos에서 사용한 MSP430의 장점으로는 wakeup time이 Mica, Mica2의 180μs에 비해서 6μs로 가능하며 전력 소모 역시 active power 일 때 15mW, sleep power 일 때 15μW로 Mica2의 33mW, 75μW에 비해 소비 전력이 훨씬 적다[53]. 또한 ATmega 128 MCU가 2.7V 이하에서 제대로 작동을 안 하는데 반해 MSP430은 1.8V까지 떨어져도 작동이 가능하다. Tmote SKY 역시 센서 노드 OS로는 TinyOS가 사용되고 있

다.

- **Maxfor TIP**

한국전자부품연구원(KETI)에서 개발한 무선 센서 네트워크 플랫폼으로 TIP가 있다[54]. TIP 3X 시리즈는 900MHz대의 RF를 사용하여 38Kb명의 데이터율을 가지며 8bit MCU로 이뤄져있다. TIP 5X 시리즈는 2.4GHz대의 RF를 사용하며 TIP50은 16bit RISC CPU로, TIP51은 Telos에서 사용되는 MSP430 MCU를 사용한다. OS로는 TinyOS를 사용한다.

- **Nano-24**

국내에서 개발된 센서 노드 중 유일하게 ETRI에서 개발한 센서 네트워크 운영체제인 Nano Qplus,.가 포팅되어 있는 플랫폼이다. Nano-24는 Crossbow의 MicaZ 보드와 유사한 구성을 가지며 RF 역시 Chipcon사의 CC2420을 사용하여 Zigbee를 지원하다. Nano-24 보드의 특징은 [그림 2-6]에서처럼 조도, IC 온도, 습도, 적외선, 가스, 초음파, 가속도 센서 등 많은 센서 노드를 제공하며 상대적으로 낮은 하드웨어 전류 소모를 목적으로 설계되었고 배터리 모니터링 기능을 가지고 있다.

- **ZigbeX Mote**

한백전자의 ZigbeX Mote의 MCU는 ATmega128 CPU를 사용하고 RF는 Chipcon사의 CC2420을 사용한다. 특징으로는 기본으로 온도, 조도, 습도 센서, RTC를 모드에 장착하고 있다. 다양한 옵션 센서 모듈을 지원하는데 기상센서, 방법센서, 홈 오토메이션 센서 모듈, 위치추적 센서, 구조 역학 센서, 바이오 센서, 자동차 센서 등의 응용 모듈을 제공한다. 센서 노드의 OS의 경우 TinyOS와 μC/OS를 지원한다.

- **UStar-2400**

UStar-2400은 Chipcon CC2420 RF 모듈을 사용하여 2.4GHz Zigbee 통신을 가능하게 하고 Atmel의 ATmega128L 프로세서를 장착하고 있다. 저전류 소모로 배터리

모니터링 기능을 제공하며 저전력 설계 보드를 사용 low-power 관리 기능을 제공한다. TinyOS와 TinyDB를 지원하며 UStar-Dev라는 emulator를 포함한 개발 환경과 센서 네트워크 응용 프로그램 등을 지원해 준다.

## 2.4 하드웨어 보안 플랫폼

임베디드 시스템의 연구과 개발, 구현이 증가하면서 점차적으로 임베디드 시스템의 보안이 중요한 관심사로 떠오르고 있다. 유비쿼터스 휴대용 기기들이 보안에서 가장 중요한 문제가 되고 있다. 유비쿼터스 휴대용 기기들은 크기가 작을 뿐 아니라 이동성 때문에 이러한 장치들은 쉽게 분실이나 도난당할 수 있는 문제가 있다. 반면에 이러한 기기들을 위한 보안 솔루션은 비용, 크기의 제한, 연산 및 배터리 자원의 사용으로 인해 제한을 많이 받는다. 따라서 가격, 크기, 전력, 프로세서 성능의 한계 내에서 보안성을 크게 향상시키는 것이 중요한 이슈가 되고 있다.

TPM(Trusted Platform Module)은 저전력, 고성능의 하드웨어 및 소프트웨어 결합을 통해 Trusted Computing 보안 목적을 구현하고 있다[58]. TPM은 Trusted Computing Group(TCG)에 의해 제정된 산업 표준 규격을 기초로 한 마이크로 컨트롤러이다. 1999년 10월, 인텔, MS, IBM, HP, 컴팩이 비영리 조직인 TCPA (Trusted Computing Platform Alliance)를 설립하고, 보안저장장치와 암호 기능을 갖춘 TPM(Trusted Platform Module)을 정의한 것이 하드웨어 보안의 시초이다. TCPA는 2001년 1월 하드웨어 시스템에 기반한 TPM 버전 1.0을 발표했다. 이 표준은 호스트에 내장된 하드웨어 모듈을 통해 불법 사용자의 데이터 수정을 방지하는 것으로 자주 사용하는 ID 인증과 데이터 암호 방식을 통해 보안 기능을 강화한 것이다. TCPA는 2003년 4월 TCG(Trusted Computing Group)로 명칭을 변경하고, 소프트웨어를 '신뢰성 있는 컴퓨팅'에서 제안한 방식과 결합한 TPM 버전 1.2 표준을 제정했다. TCG의 목표는 신뢰성 있는 컴퓨팅을 PC에서부터 이동 단말기, 서버의 각종 플랫폼으로까지 확대해 사용자의 정보자산을 보호하고, 소프트웨어 공격과

물리적인 침해로 인한 손실을 줄이자는 것이다. TPM은 장치에 부착된 작은 실리콘으로서 디지털 키나 인증서, 암호 등을 안전하게 저장하며, 가상 및 물리적 공격이 상당히 어렵다. 하드웨어 기반의 이 보안장치는 관련 소프트웨어를 포함할 수도 있고 보안 위험에 대한 정확한 측정도 가능하다. 일단 위험이 관측되면 적절한 정책을 포함시키고, 위험 부담을 맡으며, 컴퓨팅 환경을 개선하는 등 위험을 완화시키도록 할 수 있다.

TPM은 보안 키 발생기이며, 키 캐시 관리 장치로 산업 표준 암호 체계인 API를 지원한다. TPM은 암호화키를 하드웨어에서 생성, 저장, 관리하며, 시스템 플랫폼의 나머지 자원을 강화하고, 이전에는 보안되지 않은 메모리에 숨겨진 키를 갖춘 소프트웨어 암호화 알고리즘에만 의존했던 많은 애플리케이션을 저비용 고효율의 방식으로 강화한다. 데이터 및 파일의 소프트웨어 암호화만을 채택하는 임베디드 시스템에서 키는 통상적으로 보안처리 되지 않은 메모리에 저장된다. 키는 스마트카드나 USB 메모리처럼 잘못된 곳에 배치되기 쉽고 TPM보다 비용이 더 드는 제거 가능한 토큰 상에 저장될 수 있다. TPM은 저장 루트 키를 포함한 암호화 키, 승인 키 및 기타 민감한 데이터를 비휘발성 메모리에 저장하며 이러한 비휘발성 메모리는 TPM 컨트롤러 및 TPM의 불법 조작 탐지 회로에 의해 보호된다.

TPM은 다섯 가지의 주요 보안 기능을 관리한다. 하드웨어 난수발생기(RNG)를 사용하여 칩에 내장되는 한 쌍의 키를 발생시키기 위해 공개키 기능을 수행하고, 공개 키 서명, 암호화 및 복호화를 관리하여 데이터 및 디지털 비밀을 안전하게 저장할 수 있도록 한다. 이들은 부팅 시 기계의 설정 증명을 검증할 수 있는 해시(사전 작동시간 설정 정보로부터 계산된 고유 수)를 저장하고, 인식키가 TPM에서 발생되도록 익명으로 설정하는데 사용될 수 있는 승인키를 생성한다. 그리고 소유자가 TPM 기능을 켜거나 끄고, 칩을 리셋하며, 그 기능의 소유권을 보유할 수 있는 기능을 초기화하고 관리한다.

실질적인 RNG는 TPM 내부의 RSA 키 생성에 사용된다. TPM의 마이크로프로세서는 내부 기능 블록 및 나머지 시스템 자원에 대한 인터페이스를 포함하여 전 TPM의 기능과 시퀀싱을 컨트롤한다. 시스템 프로세서와 내부 TPM 메모리 사이에

서 데이터를 옮기며 암호화 엔진을 시퀀싱 한다. TPM의 RNG는 암호화 프로세서의 암호, 복호 및 키 발생 기능용 시드 키를 생성한다. 범용 시스템 프로세서에서 RSA 계산을 오프로드 함으로써 TPM은 기본적으로 시스템과 암호화 성능을 모두 향상시킬 수 있다. TPM은 해시 알고리즘을 사용하여 알려진 위임 상태에서 시스템 소프트웨어 및 하드웨어를 테스트한다. 측정 결과는 보안 처리된 비휘발성 메모리에 저장된다. 후속 측정 결과나 저장된 위임 상태 측정치와 비교되고, 변경이 이루어지면 소프트웨어 또는 하드웨어가 수정된 시스템에 경보를 발령하여 바이러스나 웜의 침입을 표시한다.

장치의 유형에 따라 임베디드 시스템은 보안 생성을 제공할 필요성이 있을 수도 있으며 다음 중의 하나 또는 전체를 프로세싱 및 저장해야 한다. 사용자 신원, 허용된 장치만이 네트워크 또는 서비스에 연결할 수 있는 네트워크 액세스, 비밀번호, PIN, 키 및 인증 등과 같은 민감한 정보의 저장, 바이러스 및 트로이의 목마로부터의 보호, 물리적 전기적 공격으로부터 하드웨어의 보호 (변경에 대한 저항), 네트워크에서 다른 장치 또는 서버와의 사이에 교환된 개인 정보 및 데이터의 무결성, 그리고 기기에 다운로드 되거나 저장된 컨텐츠가 컨텐츠 제공자가 설정한 조건에 따라 사용된다는 보장을 해야만 한다. 그리고 추가 적으로 해결해야 할 문제는 다음의 여섯 가지로 구분할 수 있다.

- 보안 알고리즘의 계산 부담 : TPM이 알고리즘 및 기타의 보안 기능 전용 저전력, 고성능 프로세서를 포함하고 있으므로, 동작 코드 개발과 관련하여 메인 CPU의 과부하와 보안 기능을 위한 별도의 프로세서 추가 중 하나를 선택할 수 있어야 한다.

- 널리 상호 운용될 수 있는 표준의 부재 : 독점적 솔루션과 비교할 때 TCG에 의해 개발되는 산업 표준 규격은 공급업체들이 상호 운용 가능한 장치를 개발하는데 도움이 되며, 많은 기업들의 경험과 지적 재산권으로부터 혜택을 받을 수 있기 때문에 필요하다.

- 이용 가능한 파워의 제한 : 특정 추산에 의하면 배터리로 동작하는 장치 상에서 보안 애플리케이션을 구동하는 경우 배터리의 수명이 절반 이하로 단축된다. 소프트웨어보다는 하드웨어에서 보안 기능을 실행하여 전력을 경제적으로 사용함으로써 TPM은 배터리의 수명을 크게 연장할 수 있도록 한다.

- 손상 또는 도난 같은 물리적 침투에 대한 취약성 : 특정 공급사의 TPM은 부당한 변경을 탐지하는 회로를 사용하여 모듈에 침입하고자 하는 시도를 탐지하고 있지만 이는 TCG 규격에서는 불필요하다. 예를 들면, 전압, 클럭 주파수, TPM 운영 환경의 기타 측면이 모니터링 되어 변경의 징후를 탐지한다. 만일 환경이 설정된 범위를 벗어나면 변경 예방 회로가 활동을 시작하여 TPM 내에 저장된 민감한 정보에 대한 액세스를 방지하도록 해야 한다.

- 단가 및 개발 기간에 관한 고려사항 : 독점적 회로 솔루션은 흔히 실제 RNG보다는 의사 RNG를 사용하기 때문에, 예를 들면, 침입에 있어서 키가 좀 더 취약해진다. 이러한 솔루션들은 흔히 소프트웨어에서만 이루어지는데 이는 컴퓨팅 측면에서도 불충분할 뿐 아니라 침입도 훨씬 쉽다. 또한 솔루션이 인증 절차를 거쳐야 할 경우 비용이 증가할 뿐 아니라 개발 속도도 느려진다. TPM이 표준을 준수하고 있으므로 독점적 솔루션에 비해 시간과 비용의 절약이 상당히 크다고 볼수 있다.

- 무선 통신 보안의 약점-모바일 기기는 보통 무선 통신을 사용하는데 이는 타인이 물리적 신호를 쉽게 엿들을 수 있으며 해커의 액세스가 용이하다는 것을 뜻한다. 무선 보안은 대부분의 모바일 기기에서 해결되어야 할 문제점이다. 무선 보안은 대부분 프로토콜에 기반을 두고 있으며, 기본적으로 소프트웨어로 구현되어 있고 철저하게 테스트되지는 않았다. 반면에 TPM은 하드웨어 기반의 보안을 제공하며, 철저한 테스트를 거친 알고리즘을 구현하고 있

다. 임베디드 시스템 보안 관리자들에게 문제로 인식되고 있는 모든 일반적인 보안 영역을 해결하기 위한 효과적인 방법을 제공함으로써 TPM은 기존은 소프트웨어만을 사용하는 솔루션에 비해 보안과 사용의 적절성이 한층 강화되었다. 현재 OEM 기업들은 산업 표준에 근거한 개방형 시스템 아키텍처로 적절하고, 인증 가능한 하드웨어 보안을 제공할 수 있도록 한다.

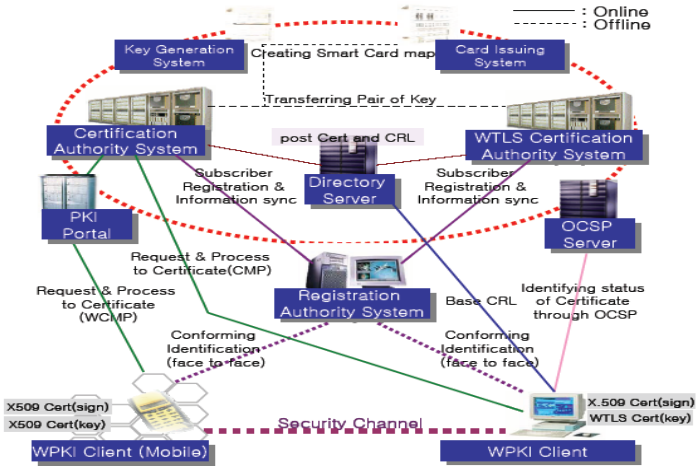
## 2.5 기존의 센서 네트워크 보안 정책

무선 센서 네트워크는 가까운 미래에 널리 사용될 수 있는 기술로서 무선 센서 네트워크 서비스 구성에 대한 연구 및 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]. 센서 네트워크에서의 보안은 암호키를 관리할 수 있는 기능 제공, 센서 환경에 적합한 경량화 된 암호 및 인증 기능 제공, 라우팅 시에 보안 기능 제공, 서비스 거부 공격에 강한 구조 등을 포함 하고 있어야 한다[2]. 따라서 센서 네트워크에서 필요한 보안 기능으로는 암호 알고리즘, 키 관리 및 보안 프로토콜, 인증 및 시큐어 라우팅이 포함되어야 하며 프라이버시와 침입 시큐어 감시 등도 추가적으로 고려를 해야 할 것이다[4]. 본 장의 4절에서는 유비쿼터스 네트워크에서 현재 진행 중인 보안 및 인증 관련 기술 연구에 대해서 기술한다.

### 2.5.1 무선 PKI(Public Key Infrastructure)

무선 네트워크는 무선 환경의 특성상 정보 노출이 용이 하고 제한된 자원을 사용해야 하기 때문에 기존의 유선 네트워크에서 사용하던 유선 PKI 구조로는 이에 적

절하게 대처하기가 어렵다. 최근에 무선 환경에 적용하기 좋은 PKI 기술을 개발하여 규격화하기 시작하였다. 관련 해외 기술 동향으로는 WAP Forum, Wireless Village, LiF, SyncML, MMS, IOP 등의 단체를 통하여 발족한 OMA(Open Mobile Alliance)가 있다. OMA는 무선 사업의 활성화와 무선 서비스의 상호연동성 보장 및 규격 표준화를 목적으로 하고 있으며, 구성은 보안, 모바일 웹 서비스, 모바일 거래 등 13개 기술 회의로 활동하고 있다. 현재 WAP(Wireless Application Protocol)은 WAP2.0으로 진화하면서 두 가지 커다란 변화를 하고 있다. 첫 번째 변화는 XHTML의 지원과 TCP/IP 기반의 프로토콜의 사용으로 유무선 인터넷 통합의 기반을 마련하고 두 번째 변화로는 기존의 유선과 무선의 게이트웨이를 통한 프로토콜 컨버전이 아닌 프록시의 역할을 하게 함으로써 통신효율성을 증대하였다. 또한 새로이 무선의 단말기와 유선의 웹 서버 사이의 보안 취약성을 해결하기 위해 TLS (Transport Layer Security)를 통한 단대단 터널링을 제안함으로써 보안 강도를 강화하고 있다[8].




[그림 2-5] 무선 PKI 구조

국내 기술 동향으로는 무선 인증서 관리 프로토콜 부분에서 유선의 CMP (Certificate Management Protocol)과 대응되는 무선 환경의 인증서 관리 프로토콜 규격을 개발하며, 무선 인증서 요청 형식 프로토콜, 식별번호를 이용한 본인확인 기

술, 무선 WAP/ME(Mobile Explorer) 인증서 프로파일 및 폐지 목록 프로파일 규격 등의 기술을 개발하고 있다[8]. 무선 PKI 기술의 각 컨퍼넌트는 다음과 같은 특징을 가지고 있다.

MS(Mobile Station)은 PDA, 모바일 폰 등의 이동 디바이스 기반의 통신 수단으로 작은 컴퓨팅 파워 때문에 공개키 알고리즘의 사용이 용이하지 못해서 암호화 알고리즘 등에 대한 코드 및 성능 최적화가 문제점으로 지적받고 있고, CP(Contents Provider)와 통신상 발생하는 보안 취약성을 극복하기 위해 종대종 보안이 지원되어야 한다.

WPKI CA(Wireless PKI Certificate Authority)는 인증서 생성을 주 기능으로 갖는다. 인증서는 각 개인이 소유하고 있는 개인키에 대응되는 공개키의 신뢰성을 주기 위해 TTP(Trusted Third Party)로 인정되는 인증 서버가 전자 서명한 형태의 메시지로 구성하는 규격으로서, 무선 인증 서버는 WTLS 인증서와 X.509 v3 인증서를 생성한다. 사용되는 알고리즘은 속도나 안전성이 무선 망 환경에 적합한 타원곡선 공개키 알고리즘을 적용한다. WTLS 인증서는 무선 WTLS 인증서 프로파일 규격에서 제시하고 있는 인증서 형식을 따라 생성하도록 하고 있다.

WPKI Portal 시스템은 등록관리 시스템과 유기적으로 운영되는 시스템으로서 MS에서 인증 서비스를 온라인으로 받기 위해 필요한 시스템으로 인증서 발급, 갱신, 폐지 등에 관련된 관리 대행의 전반적인 기능을 갖는다.

CP(Contents Provider)는 기본 웹 서비스의 확장 개념으로 MS에 콘텐츠를 제공하며, 유선단에 구축되어 X.509 무선 형식의 인증서를 사용하도록 한다. 또한 MS와 통신상 발생하는 보안 취약성을 극복하기 위해서 종대종 보안을 지원해야 한다[8].

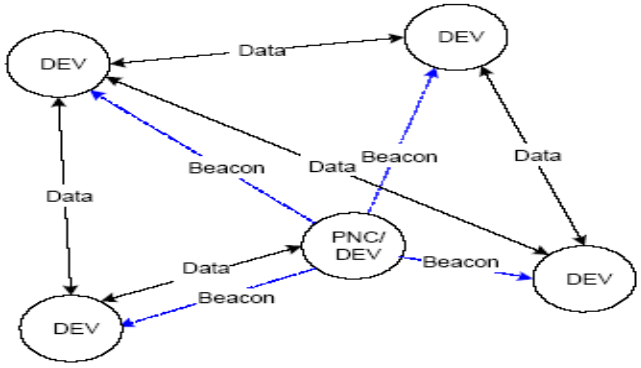
### 2.5.2 무선 LAN 보안 기술

현재 가장 많이 사용하고 있는 802.11b 무선 LAN은 처음부터 보안에 큰 관심을 두지 않았고, 공중망에서의 활용을 전제로 설계되지 않았다. 예를 들면 무선 LAN

은 브로드캐스팅 특성으로 인하여 도청 등 무선 데이터 프라이버시에 대한 취약성이 예상 되었음에도 불구하고 동적인 키 분배 방법이 없다거나, 취약한 무결성 알고리즘을 사용하여 데이터 프라이버시를 제공하지 못한다는 점이다. IEEE 802.11i(Enhanced MAC security) 작업 그룹은 최근 무선 LAN 인프라 망과 Ad-hoc망에 적용할 수 있는 새로운 형태의 보안 아키텍쳐(Robust Security Network, RSN)를 제안하고 표준화를 완료했다. RSN은 다수의 AP가 연결된 핫스팟에서 802.1x 기반 가입자 인증을 통한 네트워크 접속제어, 보안 세션 관리, 패킷당 키 관리, 그리고 새로운 암호 알고리즘 도입을 통한 무선접속 구간 보안을 강화하는데 이용된다. RSN은 WEP(Wired Equivalent Privacy)의 단점을 보완하기 위해 단기적인 관점에서 보안 문제점을 소프트웨어적으로 개선하는 TKIP(Temporary Key Integrity Protocol)을 사용하는 방식이다. RSN은 핫스팟에서 802.11과 802.11i를 지원하는 무선 LAN AP들이 공존하는 환경에서, 802.1x를 이용하여 가입자 인증 및 키 관리 메커니즘과 빠르고 안전한 로밍 보안 프레임워크를 제사한 새로운 형태의 보안 구조이다.

IEEE 802.15.3 WPAN의 보안 메커니즘은 PNC(Piconet Coordinator)와 장치 사이, 장치간의 보안 맴버쉽/릴레이션쉽을 바탕으로 데이터에 대한 암호화를 지원하지만, 보안 맴버쉽/릴레이션쉽을 확립하는 장치 인증과 같은 절차는 지원하지 않는다. 또한 데이터 프레임과 Command 등을 보호하기 위해 128bit AES(Advanced Encryption Standard) 방식의 대칭키 암호화 알고리즘을 사용한다. 이때 어떤 하나의 장치에서 다른 장치로 전송되는 모든 키는 Key Request Distribute Key Protocol에 의한 방법으로 전송된다. 데이터, Beacon, Command에 대한 무결성 검사는 알맞은 키에 의해 이루어진다. Beacon에 대한 무결성 검사가 실패하면, 디바이스와 PNC는 동기화된 보안 상태를 가지지 못한다. 한번 사용된 메시지가 사용되는 것을 막기 위해 Beacon 내에는 Strictly-Increasing Time Token이 포함된다. 따라서 디바이스는 현재의 Time Token 보다 작거나 같은 Time Token을 Beacon에서 찾으면 그 Superframe을 버린다. 또 Device는 현재 Superframe의 Beacondptj 얻게 되는 Current Time Token을 사용하여 Superframe 동안에 보내고 받는 모든

메시지를 보호하며, Last Valid Token을 사용하여 Beacon의 보안이 훼손되지 않았다는 것을 보장한다.




[그림 2-6] IEEE 802.15.3 WPAN의 Piconet 구조

보안 문제는 유선 네트워크에서도 존재하고 있지만, 모바일 Ad-hoc 네트워크는 무선 인터페이스를 사용하기 때문에 유선 네트워크에 비해 훨씬 더 많은 위험에 노출되어 있다. 따라서 기본적인 Ad-hoc 네트워크의 보안 요구 조건은 다른 통신 네트워크에서 요구되는 것과 동일하지만 모바일 Ad-hoc 네트워크에서는 노드의 신분이 서로에게 불확실한 경우가 많으며, 멀티 홉 방식을 이용한 라우팅을 수행할 경우, 악의적인 중간 노드에 의해 데이터의 무결성 및 비밀성 문제가 발생할 수 있다. 특히 매체를 신뢰할 수 없는 상황에서 암호를 사용하는 경우가 많아서, 암호키에 크게 의존하게 된다. 또한 모바일 Ad-hoc 네트워크 환경은 모든 노드들이 분산되어 있고 어떠한 고정된 기반 구조도 없으며 모든 노드가 공평하게 역할을 나눠 갖는다는 특징을 갖는다. 더욱이 보안 문제가 확실히 해결되다 보면 노드의 처리 문제가 발생되어 노드와 네트워크 전체에 심각한 부하를 주게 되므로 이러한 trade-off를 고려한 모바일 Ad-hoc 네트워크에 적합하게 구현된 알고리즘 키 분배 및 인증 프로토콜의 개발이 현실적으로 가장 필요하다. 즉 키 사이에 신뢰할 수 있는 관계를 형성하고 이를 모바일 Ad-hoc 네트워크 전반에 분배하는 것이 주요 과제가 된다[1].

### 2.5.3 TinySec

TinyOS는 미국 버클리 대학에서 개발한 현재 가장 널리 쓰이고 있는 컴포넌트 기반, 이벤트 구동 방식의 센서 네트워크 운영체제로서, 제한된 자원을 가진 작은 크기의 센서 노드에서 효율적인 자원의 사용과 프로세싱 동시성을 지원해 준다[13][67]. 센서 네트워크의 제약 사항과 새로운 요구 사항을 만족시키기 위해서 TinyOS 는 다음과 같은 특징들을 가지고 있다. 첫째, 하드웨어 추상화(Hardware Abstraction)부터 고수준(High Level) 소프트웨어까지 모두 재사용 가능한 컴포넌트들로 구성되어 있다. 각 컴포넌트는 입/출력 커맨드(Command)와 입/출력 이벤트(Event)로 구성된 인터페이스(Interface)를 통해 다른 컴포넌트들과 상호 작용한다. 둘째, 상태 머신 프로그래밍 모델을 기반으로 함으로써 전체 시스템은 여러 상태 머신들로 구성되어 있다. 여기서, 각각의 컴포넌트가 해당 상태를 나타내며, 컴포넌트의 커맨드 또는 이벤트 핸들러들은 한 상태에서 다른 상태로 빠르게, 그리고 적은 부가처리와 논-블럭킹의 특성을 가지며 전이시킨다. 셋째, 제한된 자원만을 가진 노드들로 구성된 센서 네트워크의 가장 큰 문제인 저전력 소모를 위해서 TinyOS는 CPU가 사용되지 않을 동안 휴지 상태로 들어가게 함으로써 불필요한 전력 소모를 줄인다. TinyOS의 핵심 OS 코드는 4000바이트 이하이고, 데이터 메모리는 256 바이트 이하이며, 이벤트 기반 멀티태스킹을 지원하며, 내장형 네트워크를 위한 프로그래밍 언어로는 nesC가 사용된다[40].

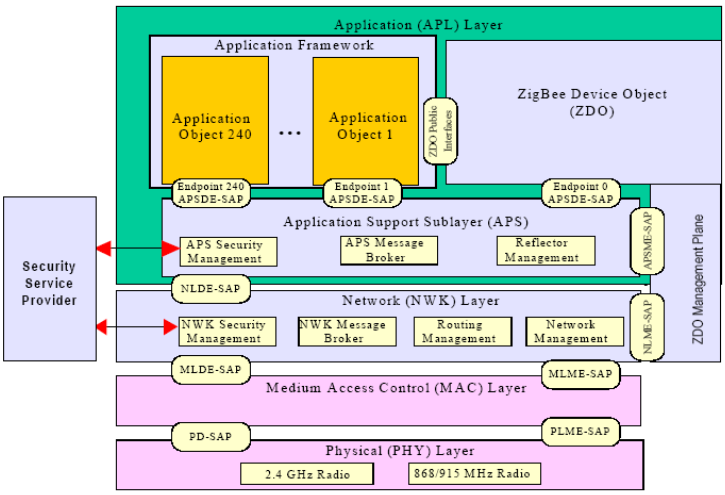
보안으로는 정책으로는 키 관리 및 접근 제어 등의 암호 서비스를 제공하기 위한 연구로는 센서 노드용 운영체제로 개발된 TinyOS에서 구동 가능하도록 하는 TinySec이 있다[3]. TinySec에서는 링크 보안 기술을 적용한 간단한 접근제어와 메시지 위·변조를 막을 수 있는 무결성 기능과 비합법적인 노드에 의한 센서 정보의 해석을 방지하는 기밀성 기능을 제공한다. 하지만 상대적으로 짧은 IV(Initial Vector)를 사용하기 때문에 평문이 공격자에게 쉽게 노출될 수 있는 약점이 있다.

### 2.5.4 Zigbee Alliance Security

산업체를 중심으로 하는 Zigbee Alliance는 저전력 무선 센서에 다양한 응용이 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 불구하고, 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.

또한 Zigbee 보안 정책에서는 마스터키를 안전하게 공유하고 있다는 가정을 기반으로 안전한 통신을 원하는 각 객체 간에 링크키를 교환하게 된다. 따라서 마스터키는 링크키를 교환하기 위하여 사용하고 마스터키를 이용하여 분배된 링크키를 이용하여 통신을 하게 된다. 본 논문에서는 주 사용키와 보조 저장키를 사용하는 더블키 정책과는 차이를 보이고 있다. Zigbee 보안에서 두 개의 키는 사용되는 곳이 차이가 있으며 교체 및 관리 알고리즘이 없기 때문에 하나의 마스터키가 전복이 될 경우 네트워크 전체에 영향을 줄 수 있다. 또한 안전한 전송을 원하는 객체가 링크키를 생성하여 교환을 해야 하기 때문에 유비쿼터스 센서 네트워크의 센서 노드와 같이 하드웨어 제약이 많은 곳에서 사용하기에는 제약이 발생하게 된다.




[그림 2-7] Zigbee 스택 구조

### 2.5.5 SPINS

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-TESLA 와 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 것 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.

SPINS에서 제시하는 프로토타입은 소규모의 전력으로 구동되는 노드와 보다 강력한 자원을 갖고 있는 베이스 스테이션으로 구성된다. <표 2-1>과 같이 제한된 자원을 갖는 센서 네트워크상에서 안전 신뢰성 있는 서비스를 제공하기 위한 연구의 일환으로 SPINS가 수행되었다. SPINS의 보안 요구 사항과 보안 인프라를 제시

하기 전에 실제적인 요구사항과 시스템 구성도를 정의해야 한다. 이러한 정의는 일반적인 센서 네트워크상에서 일반적인 요구 사항을 충족하기 위한 목적을 갖는다. 일반적으로 무선망을 이용한 센서 노드 통신 방법으로 브로드캐스팅 방식이다. 이것은 센서 네트워크 서비스 특성상 최소한의 자원 소모에 적절한 보안적 요구사항을 만족하기 위한 적절한 보안 수준을 제공하는 것이다.

<표 2-2> SPINS을 수행하기 위한 센서 네트워크 사항

| 항목 | 사양 |
|---|---|
| CPU | 8bit, 4MHz |
| 저장 공간 | 8Kbytes instruction flash<br>512 bytes RAM<br>512 bytes EEPROM |
| 통신 | 916 MHz radio |
| 대역폭 | 10 Kbps |
| 운영 체제 | TinyOs |
| 운영 체제 코드 공간 | 3500 bytes |
| 사용 가능한 코드 공간 | 4500 bytes |

전형적인 SmartDust 센서 네트워크는 외부 망과의 연계를 위해 하나 이상의 베이스 스테이션을 갖는다. 이러한 베이스 스테이션은 충분한 자원을 갖고 있으며 많은 센서 노드와 통신 관리의 중심적 역할을 수행한다. 일반적으로 센서 네트워크를 위해 운영되는 센서 노드는 안전하지 않은 위치에 설치된다. 따라서 각 노드에 대한 신뢰성을 보장 받을 수 없기 때문에 한 노드의 보안 노출이 다른 노드에 영향력을 미치지 않도록 보안 사고의 최소화가 절대적으로 필요하다. 각 노드는 안전하지 않은 무선망에서 공격자의 도청이 언제든지 가능하며 메시지의 재사용 공격에 매우 취약하다. 그러나 베이스 스테이션은 외부 망과의 게이트웨이 및 센서 네트워크의 중심에 있으므로 언제나 안전해야 하며 각 노드는 이러한 베이스 스테이션과의 통신을 위해 초기 설치 시 마스터 키 값을 할당 받는다. 각 노드에 설치된 마스터 키는 베이스 스테이션과 공유되는 대칭키로서 향후 사용되는 모든 키 생성을 위한

SEED로 사용된다[8].

SNEP(Secure Network Encryption Protocol) 기술은 다음과 같은 보안 기능을 제공한다. 의도된 수신자만이 데이터를 소유할 수 있도록 데이터를 비밀키로 암호화하며 제3자가 암호 메시지에서 원래 메시지를 추론 할 수 없는 오만 기능을 보장한다. 암호화를 위해서는 Counter Mode를 적용하여 비밀성을 보장한다. 그러기 위해서 송수신자 사이에 shared counter를 사용한다. 따라서 같은 메시지가 매번 다르게 암호화된다. 의도된 송신자가 정말로 해당 데이터를 보냈는지 검증하기 위해서 공유된 키를 기반으로 MAC을 사용한다. MAC에 카운터 값을 포함하여 공격자에 의해 재사용 공격이 이루어질 경우 카운터 정보로 판별한다. SNEP에서는 일반적인 RC5 구현에서 대칭키 암호화 알고리즘을 이용하여 다양한 목적의 키 값을 유추한다. 마스터 키를 기반으로 암호화를 위한 암호화 키, MAC 값 생성을 위한 키 값, 랜덤키 값 등을 생성한다.

센서 네트워크상에서 TESLA의 문제점과 μ-TESLA에서의 해결방법은 다음과 같다. TESLA 기술은 초기 패킷에 전자 서명을 포함하여 인증하는 방식으로 시스템의 부하가 가중되지만, μ-TESLA에서는 동기화 메커니즘만 사용여 인증 절차를 수행한다. TESLA 기술은 각 패킷에 사용된 키를 노출하여 송수신하므로 많은 전력이 소모되지만 μ-TESLA에서는 주기별로 한 번씩만 키를 노출시켜 자원 효율을 극대화 한다. TESLA 기술은 단방향 key 체인을 센서 노드에 저장, 관리하므로 고비용이 소요되지만, μ-TESLA에서는 인증된 센서의 수를 제한하므로 효과적인 관리가 가능하다[8].

### 2.5.6 센서 네트워크에서 공개키 기반의 보안 연구

유비쿼터스 센서 네트워크의 제한적인 센서 노드의 환경으로 처리량이 많고 처리시간이 상대적으로 더 긴 공개키 기반의 보안 정책을 적용하기는 힘들다. 따라서 현재 유비쿼터스 센서 네트워크 보안 정책의 대부분은 대칭키 기반의 보안 정책을

이용하여 연구되고 있다. 하지만 점차 하드웨어의 발전과 경량화 된 공개키 기반의 보안 알고리즘들이 연구 되면서 최근 들어 유비쿼터스 센서 네트워크에서도 공개키를 이용한 보안 정책들이 활발히 이루어지고 있는 실정이다.

타원 곡선 암호(ECC:Elliptic Curve Cryptographic) 알고리즘은 기존의 공개키 알고리즘에 비하여 상대적으로 적은 처리량으로 같은 보안 효과를 볼 수 있는 장점으로 최근 센서 네트워크에 적용하려는 연구가 늘어나고 있다. TinyOS 환경에서 타원 곡선 암호 알고리즘을 이용하려는 연구가 진행 중이며[79] Haodong Wang[78]은 센서 네트워크에서 타원 곡선 암호 기반의 접근 제어를 연구 중에 있다. Lejla Batina[80]는 무선 센서 네트워크를 위한 저가의 타원 곡선 암호를 연구 중에 있으며, 유비쿼터스 센서 네트워크를 위한 Leif Uhsade[81]의 연구, 무선 센서 네트워크에 구현하기 위한 연구[82] 등이 현재 진행 중에 있다.

또한 공개키 기반의 보안 정책을 센서 네트워크에 적용시키기 위한 연구들도 진행 중에 있다. Gaubatz[83]의 센서 네트워크에서 공개키 보안에 관한 연구, Wander[84]의 무선 센서 네트워크를 위한 공개키 암호의 에너지 분석에 관한 연구, Du[85]의 무선 센서 네트워크를 위한 공개키 분배 연구, Murphy[86]의 무선 센서 네트워크를 위한 공개키 암호를 이용한 하드웨어·소프트웨어 구현과 같은 연구들이 진행되고 있다. 또한 TinyOS에 대표적인 대칭키 기반의 TinySec 보안정책과 대조적으로 TinyOS에서 공개키 기반의 보안 정책을 사용할 수 있도록 하는 TinyPK[87] 기술도 연구 중에 있다.

# 제 3 장 유비쿼터스 센서 네트워크를 위한 경량 보안 프로토콜

본 논문에서 제안하는 유비쿼터스 센서 네트워크를 위한 경량 보안 프레임워크는 인증 및 키 교체 관리 프로토콜과 센서 네트워크 감시 프로토콜로 이루어져 있다. 두 가지 프로토콜 중에서 본 3장에서는 유비쿼터스 센서 네트워크 환경에 맞게 경량화 된 인증 및 키 교체 관리 프로토콜에 대해서 제안을 한다. 본 논문에서 제안하는 인증 및 키 교체 관리 프로토콜은 대칭키 기반의 프로토콜과 공개키 기반의 프로토콜의 두 가지 프로토콜로 구분 할 수 있다. 제안하는 두 가지 방식의 프로토콜 중 하나의 프로토콜과 4장의 설명하는 센서 네트워크 감시 프로토콜과 구성하여 유비쿼터스 센서 네트워크를 위한 경량 보안 프레임워크를 구성하게 된다.

3.1절에서는 대칭키 기반의 더블키를 이용한 인증 및 키 교체 관리 프로토콜에 대해서 설명을 하고 3.2절에서는 공개키 기반의 공개키를 이용한 인증 및 키 교체 관리 프로토콜에 대해서 설명한다. 3.3절에서는 제안하는 보안 프로토콜의 안전성에 대하여, 3.4절에서는 제안하는 두 가지 방식의 보안 프로토콜 중에 어떠한 기준으로 보안 프로토콜을 선택하여 센서 네트워크 감시 프로토콜과 경량 보안 프레임워크를 구성할 것인가에 대한 논의를 한다.

## 3.1 더블키를 이용한 보안 프로토콜

유비쿼터스 센서 네트워크를 위한 경량 보안 프레임워크는 서버와 센서 노드간의 인증 및 키 교환 관리 프로토콜과 센서 네트워크 감시 프로토콜로 이루어져 있다. 본 절에서는 대칭키 기반의 더블키를 이용한 인증 프로토콜에 대하여 설명한다. 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 간

의 인증 및 보안 정책에 중점을 두고 설계하였다.

본 논문에서 새롭게 정의하여 사용하는 더블키(Double-Key) 정책은 기존의 대칭키 기반의 보안 정책의 단점을 해결하기 위해서 제안되었다. 대칭키 기반의 보안은 대칭키의 분배 문제가 가장 중요한 해결 사항이다. 또한 안전한 통신을 위해서는 주기적인 키 교체가 필수적인데 네트워크 환경에서는 구현하기가 쉽지가 않다. 이를 해결하기 위한 방법으로 각각의 다른 객체에서 같은 키를 생성해 내는 연구가 진행 중이다. 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 명칭 한다. 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 두 개의 키가 존재 하게 된다.

인증 과정에서 두 개의 대칭키를 서버에서 생성을 해서 인증을 하는 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 모듈에서 전달 받아서 MCU의 레지스터에 저장을 하게 된다. 서버에서 받은 메인키와 보관키는 MCU의 레지스터에서 Flash 메모리로 저장을 하게 된다. 그 중 메인키는 보안 모듈에서 계속해서 사용을 하게 되며, 보관키는 키 교체

프로토콜이 동작하기 전까지는 계속해서 센서 노드의 MCU Flash 메모리에 보관을 하게 된다. 일반적인 센서 노드에서는 외부에서 Flash 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.

제안하는 더블키를 이용한 경량 보안 프로토콜 정책을 사용하였을 경우 발생하는 다음과 같은 장점이 있다. 첫째, 각 개체 간 인증을 할 경우에 기존의 인증 방식을 이용하여 구현 할 경우는 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 노드의 하드웨어적인 증가가 필요 하게 된다. 본 논문의 더블키를 이용한 보안 프로토콜은 경량화 된 인증 프로토콜을 제안하여 센서 노드에는 상대적으로 크기가 작은 모듈이 포함되기 때문에 하드웨어적인 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 이용하여 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 소

비를 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 저전력으로 운영 할 수가 있다.

### 3.1.1 더블키 보안 프로토콜의 구조

공개키 기반의 암호화 정책은 연산량이 많고 포함하고 있어야할 모듈의 크기가 크기 때문에 센서 노드의 하드웨어적인 제약이 많은 유비쿼터스 센서 네트워크에서는 사용하기 힘들다. 그래서 대부분의 보안 정책은 대칭키 기반의 암호화 정책으로 설계한다. 하지만 대칭키 기반의 암호 알고리즘의 경우 대칭키의 분배가 가장 어려운 부분이다. 일부 연구에서는 대칭키가 안전한 경로로 분배 되었다고 가정 하에 대칭키 기반의 보안 정책을 설계하기도 한다. 대칭키의 분배가 어렵기 때문에 센서 노드와 관리 서버에서 같은 키 생성 알고리즘을 사용하여 같은 대칭키를 생성해서 주기적으로 교체하는 방식을 제안하기도 한다. 하지만 이러한 경우에도 제약이 많은 유비쿼터스 센서 네트워크에서 센서 노드에 실제 적용하기는 힘들다.

본 절에서 이러한 문제점을 해결하기 위해서 대칭키 기반의 안전한 대칭키 분배 및 인증 그리고 키 교체 알고리즘을 제안한다. 유비쿼터스 센서 네트워크에서 센서 노드와 중앙 서버에 대한 정의와 각각의 노드가 처리해야 할 내용은 다음과 같이 정의 할 수 있다. 중앙 처리 서버는 센서 노드로부터 오는 모든 데이터를 전송 받아서 처리, 분석 및 저장 작업을 한다. 중앙 처리 서버는 하드웨어의 제약에서 비교적 자유로운 편이며 센서 노드 및 센서 네트워크를 관리한다. 센서 노드는 주변 환경의 데이터를 수집하여 중앙 처리 서버에 데이터를 보내는 주체이며 또한 중앙 관리 서버에서 요청되는 명령어를 이행하도록 한다. 센서 노드는 하드웨어적으로 제약이 많고 작동하는 활동 주기가 정해져 있다.

이렇게 정의된 구조에서 센서 노드와 중앙 처리 서버의 보안 정책 관계는 다음과

같이 구분 할 수 있다. 중앙 처리 서버는 센서 노드로부터 온 데이터를 해독하여 데이터를 분석한다. 센서 노드들에 대한 인증 권한과 해당 센서 노드의 보안 상태, 암호 알고리즘 등을 관리한다. 필요에 따라서는 센서 노드를 네트워크에 가입 및 탈퇴를 시켜야 하며 해당 센서 네트워크가 공격자로부터 공격을 받을 시에는 이를 발견하고 경고를 해야 한다. 센서 노드는 수집된 데이터를 중앙 처리 서버에서만 가독할 수 있도록 암호화 작업을 하며, 관리자 또는 중앙 처리 서버의 요청에 의해서 일부 보안 관련 데이터를 수정 할 수 있다. 또한 센서 노드 내에서 사용하는 대칭키에 대한 관리를 해야 한다.

제안하는 더블키를 이용한 인증 프로토콜은 센서 노드와 중앙 관리 서버의 이러한 관계를 기반으로 해당 노드에서의 보안 정책을 정의 하였다. 각 보안 정책의 특징은 다음과 같다.

- 인증 프로토콜 : 인증은 중앙 처리 서버와 센서 노드간의 인증을 지원한다. 센서 노드는 초기화가 되고 유비쿼터스 센서 네트워크에 가입하기 전에 중앙 처리 서버와 인증 작업을 한다. 인증이 끝나기 전까지 센서 노드는 어떠한 작업도 시작하지 않도록 한다. 센서 노드와 중앙 처리 서버간의 인증이 완료가 되면 해당 센서 노드는 중앙 처리 서버가 관리하는 유비쿼터스 센서 네트워크에 포함이 되어 정상적인 작동을 시작하게 된다.

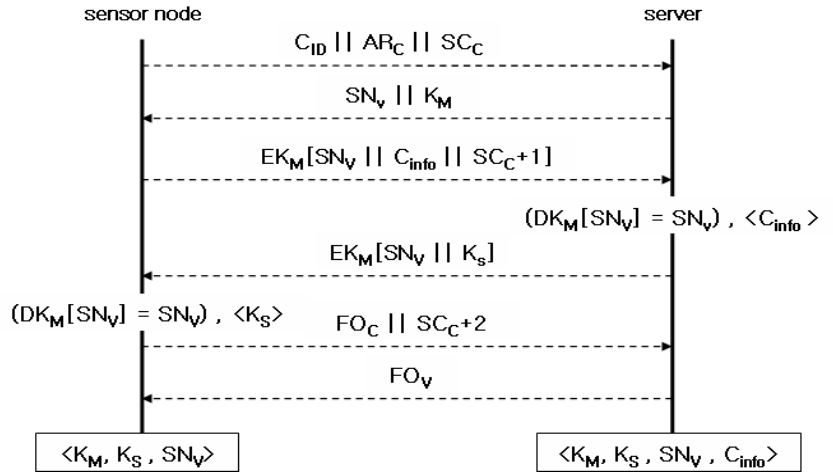
- 키 교체 및 관리 프로토콜 : 센서 노드와 중앙 처리 서버는 각각 두 개의 대칭키를 보관하게 된다. 이는 초기 인증 프로토콜에 의해서 분배가 된다. 그 중 하나의 키는 보관을 하고 하나의 키만으로 암호화 및 복호화에 사용한다. 주기적으로 혹은 중앙 처리 서버의 판단에 의해서 키의 교체가 필요할 경우 센서 노드와 중앙 처리 서버는 기존의 사용하던 메인키를 삭제하고 보관키를 메인키로 사용하게 되며, 이때 다시 보관용 키를 중앙 처리 서버로부터 받아서 각각 다시 저장하게 된다.

중앙 처리 서버의 경우 키 생성 알고리즘을 가지고 있으며 또 관리가 가능해야 한다. 또한 키 교체시기를 판단해야 한다. 센서 노드에서는 단지 인증을 받은 후에 서버로부터 받은 키를 보관을 하고 중앙 처리 장치의 신호에 의해서 키를 교체하기만 하면 된다. 이러한 관계는 중앙 처리 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 적어진다는 장점이 있다.

더블키 보안 프로토콜은 2개의 프로토콜로 이루어져 있다. 3.1.2절에서는 더블키 인증 프로토콜에 대해서, 3.1.3에서는 키 교체 및 관리 프로토콜에 대하여 설명한다.

### 3.1.2 더블키 인증 프로토콜

유비쿼터스 센서 네트워크의 센서 노드의 환경을 고려해서 센서 노드가 부담해야 하는 처리량을 줄이기 위해 설계한 더블키 인증 프로토콜의 경우 해당 프로토콜이 성공적으로 끝나기 위해서는 다음과 같은 여섯 가지 사항을 만족해야 한다. 첫째, 센서 노드는 인증을 받기 전까지 어떠한 동작도 하지 못하게 해야 한다. 둘째, 서버와 인증을 원하는 센서는 센서 네트워크 관리자에 의해서 센서 네트워크에 포함시키기로 결정한 센서만이 가능하다. 셋째, 인증을 원하는 센서와 서버간의 인증은 서버에서 결정한 시간에만 가능하다. 넷째, 인증 프로토콜 진행 시 서버와 센서 노드는 최단 거리에서 실행 하며 무선 혹은 유선을 사용하도록 한다. 다섯째, 인증 프로토콜은 정해진 시간 안에 종료가 되어야 인정을 한다. 여섯째, 위의 상황에서 한 가지 이상 제외될 경우 인증을 무효화를 시킨다. 이러한 초기 제약 조건을 만족하였을 때 서버와 센서의 인증 프로토콜은 다음 [그림 3-1]과 같다.




[그림 3-1] 서버와 센서 노드간의 경량 인증 프로토콜

[그림 3-1]에서 사용 하는 기호의 의미는 다음과 같다.

$C_{ID}$ = 센서 ID

|| = 이어 붙여 쓰기

ARc = 센서 노드에서 보내는 인증 요청 신호

SCc = 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 : $C_{ID}$ || ARc || SCc

(2) 서버 → 센서 노드 : $SN_V$ || $K_M$

(3) 센서 노드 → 서버 : $EK_M$ [ $SN_V$ || $C_{info}$ || SCc+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 : $FO_C$ || SCc+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($C_{ID}$ || ARc || SCc). 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 받더라도 서버와 센서 노드 간의 인증만 이루어질 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($SN_V$ || $K_M$). 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 완료하기는 어렵다. 만일 공격자가 대칭키를 복사 하더라도 해당 센서 노드와 서버간의 대칭키이기 때문에 다른 센

서 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($EK_M[SN_V||C_{info}||SCc+1]$). 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($DK_M [ SN_V ] = SN_V$). 다른 키를 이용하거나 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($EK_M [ SN_V || K_S ]$).

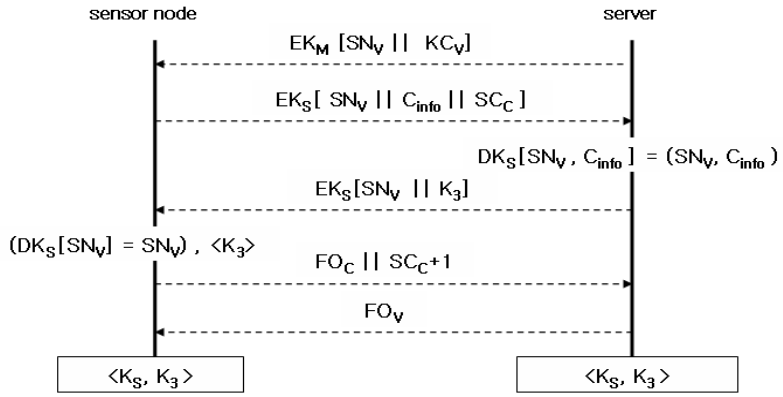
다섯 번째 단계에서는 센서 노드에서는 첫 번째 암호키로 복호화 작업을 하고 복호화 작업을 해서 얻은 시큐어 넘버와 저장을 해둔 시큐어 넘버를 확인하고 일치할 경우 두 번째 마스터키를 저장을 한다($DK_M [ SN_V ] = SN_V$). 작업이 끝나면 인증 서버에 종료 신호를 보내다($FO_C || SCc+2$).

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

### 3.1.3 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 하드웨어적인 제약 때문에 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.




[그림 3-2] 센서 노드와 서버간의 키 교체 프로토콜

본 절에서는 5장에서 설명하게 될 유비쿼터스 센서 네트워크 보안 관리자(uSNSM)의 판단이나 관리자가 정해 놓은 키 교체 주기 마다 센서 노드와 서버간의 안전한 키 교체 알고리즘을 제안한다. 교체 주기는 보안 정책에 맞도록 기간을 정할 수가 있으며 uSNSM에서 판단으로 키 교체가 가능하도록 설계 하였다. 안전한 키 교체가 이루어지기 위해서는 다음과 같은 조건을 만족해야 키 교체가 이루어지도록 설계를 하였다. 첫째, 키 교체의 경우 센서에서는 요청을 하지 못하며 서버에서 교체 신호를 보낼 경우에만 가능하다. 둘째, 교체 프로토콜은 정해진 시간 안에 종료가 되어야 허용된다. 셋째, 위의 해당 조건에 제외 되는 경우가 발생할 경우는 키 교체를 무효화 한다. [그림 3-2]는 서버와 센서 노드간의 키 교환 프로토콜을

나타낸 것이다. [그림 3-2]에서 사용하는 기호의 의미는 다음과 같다.

$C_{ID}$ = 센서 ID

$KC_V$ = 서버에서 센서 노드에서 보내는 키 교체 신호

|| = 이어 붙여 쓰기

SCc = 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 : $EK_S [\ SN_V \ || \ C_{info} \ || \ SCc\ ]$

(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 : $FO_C \ || \ SCc+1$

(5) 서버 → 센서 노드 : $FO_V$

첫 번째 단계에서 uSNSM이나 관리자에 의해서 키 교체 프로토콜이 실행이 되면, 먼저 서버에서 해당 센서 노드에 키 교체를 통보하는 신호를 보내게 된다($EK_M$ [ $SN_V$ || $KC_V$ ]). 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($EK_S$ [ $SN_V$ || $C_{info}$ || $SCc$ ]). 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 을 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($DK_S$ [ $SN_V$ || $C_{info}$ ] = ($SN_V$ , $C_{info}$)). 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($EK_S$ [ $SN_V$ || $K_3$ ]). 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 복호화 하고 비교를 하기 때문에 안전하게 분배가 가능해 진다.

네 번째 단계에서 센서 노드에서 받은 정보가 일치 하게 되면($DK_S$ [ $SN_V$ ] = $SN_V$) 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($FO_C$ || $SCc+1$) 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 프로토콜

의 경우 교체 시간을 제한하고 프로토콜 실행 중 계속적으로 데이터를 비교하면서 교체가 이루어지기 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.

## 3.2 공개키를 이용한 보안 프로토콜

일반적인 네트워크 보안에서는 공개키를 기반으로 하는 보안 정책을 사용한다. 공개키를 이용한 보안 정책은 암호키를 공개하기 때문에 누구나 키를 사용할 수 있지만 개인키는 공개가 되지 않기 때문에 인터넷이나 분산 환경 등에서 주로 사용한다. 하지만 공개키를 이용하는 보안 정책은 인증을 받고, 인증을 한 개체에 대한 각각의 다른 키가 존재하기 때문에 일대일이 아닌 일대다의 관계에서는 하나의 개체가 스스로 관리해야 할 키의 종류는 점차적으로 증가하게 된다. 이러한 점은 메모리 및 전력의 제약이 비교적 적은 컴퓨팅 환경에서는 크게 문제가 되지 않는다. 하지만 유비쿼터스 센서 네트워크와 같은 센서 노드의 제약이 많은 환경에서는 키의 저장 및 관리가 커다란 문제로 부각되고 있다. 또한 공개키 보안 정책은 암호화·복호화 연산의 연산량이 대칭키에 비하여 많으며, 연산을 위하여 포함하고 있어야할 모듈의 종류와 크기가 상대적으로 많기 때문에 실제 센서 네트워크에서는 공개키 보안 정책 보다는 대칭키 방식의 보안 정책이 대부분 제안되고 있다. 그렇지만 키의 관리에 대한 문제점과 경량화 된 암호화 알고리즘 개발 문제만 해결이 된다면 대칭키 방식의 암호 정책보다는 공개키 방식의 암호 정책이 키 분배 등에 더 많은 장점이 있기 때문에 유비쿼터스 센서 네트워크에서도 적합한 암호 정책이 될 것이다. 더욱이 센서 노드의 하드웨어 기술 발전하게 된다면 현재의 대칭키 기반의 센서 보안 정책이 아닌 공개키 기반의 센서 보안 정책이 주를 이룰 것으로 예상된다. 이러한 내용은 3.3절에서 공개키 기반의 보안 정책과 하드웨어 플랫폼과의 관계에

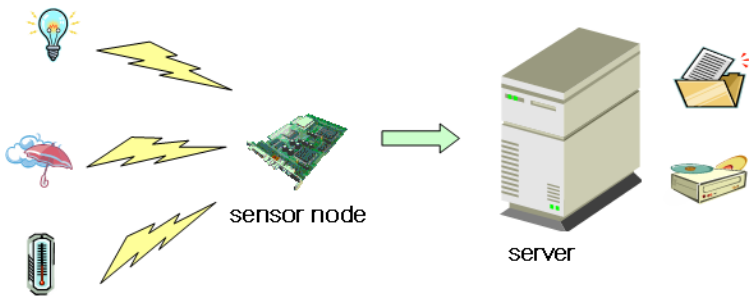
대하여 설명한다.

최근 센서 네트워크 분야에서 센서 노드에 공개키 보안 정책을 적용하기 위한 연구들이 시작되고 있다. 아직까지는 실제 센서 노드에 적용시키는 정책 보다는 접근 방법, 기술, 경량화에 맞추어 연구가 진행 중에 있다. 본 장에서 최근 연구가 진행 중인 센서 네트워크를 위한 공개키 기반 보안 기술을 실제 센서 노드에 적합하게 사용할 수 있도록 경량화와 저전력에 중점의 공개키 보안 정책을 제안한다. 제안하는 공개키를 이용한 인증 프로토콜은 기존의 공개키 보안 정책을 센서 네트워크에 맞도록 수정하여 센서 노드의 하드웨어 환경에서 인증 및 보안 프로토콜을 사용할 수 있도록 하는 공개키 기반의 보안 정책이다. 본 논문에서 제안하는 공개키를 이용한 경량 보안 프로토콜의 특징은 다음과 같다. 첫째, 기존의 공개키 보안 정책은 각 노드에서 키 관리를 해야 하기 때문에 유비쿼터스 센서 네트워크 환경에는 적합하지 않기 때문에 센서 노드의 하드웨어 발전이 되기까지는 구현하지 못하였다. 하지만 제안하는 공개키 기반의 보안 프로토콜은 센서 노드에서 관리해야 할 키의 종류가 적어서 다량의 키를 관리를 하거나 보관할 필요가 없게 된다. 둘째, 센서 노드에서는 단지 서버에서 공개하는 키를 받아서 사용만 하면 되기 때문에 센서 노드 자체에서 키 생성을 하거나 인증을 위해서 인증 서버에 계속 접속을 하거나 하는 작업을 할 필요가 없다. 따라서 공개키 보안 모듈의 경량화가 가능해 진다. 셋째, 센서 노드는 단지 주변 환경의 데이터를 수집해서 데이터를 전송을 하는 개체이기 때문에 데이터를 암호화만 하면 될 뿐 복호화의 작업을 할 필요는 없게 된다. 따라서 공개키 암호 정책의 경량화가 가능하며 연산의 량도 줄어들게 된다. 이러한 방식은 센서 노드에서 서버와 양방향 통신이 아닌 일반적인 데이터 전송을 하는 센서군에서 인증 및 암호화 작업이 가능할 수 있게 한다. 넷째, 제안하는 공개키 기반의 인증 프로토콜은 센서 노드에 인증 및 공개키 관리의 부담이 줄어들고 복호화 작업을 제외시킴으로 저전력의 효과를 볼 수 있다. 이러한 점은 기존의 공개키 보안 정책 보다 상대적으로 센서 노드의 연산량 감소로 이어지기 때문에 저전력의 효과를 기대할 수 있다.

### 3.2.1 공개키 보안 프로토콜의 구조

공개키 보안 프로토콜에 대해서 설명하기 전에 앞서 유비쿼터스 센서 네트워크 환경 및 각 개체에 대해서 정의하면 다음과 같다. 센서 노드는 주변의 환경 데이터를 수집하여 중앙 처리 서버에 전송을 하는 것을 담당한다. 중앙 처리 서버에서는 센서 노드로부터 수집된 데이터를 전송 받아서 처리, 가공, 저장을 담당한다.

[그림 3-3]과 같은 환경에서 센서 노드는 데이터를 전송하며 서버는 전송 된 데이터를 처리하는 작업만 담당하기 때문에 기존의 인증 방식과 다른 개념의 인증 방식으로의 접근이 필요하다. 기존의 인증 방식은 양방향 인증으로 서로 상대 개체를 인증 하는 방식이지만 위의 [그림 3-3]과 같은 환경에서는 종속적인 인증이 필요하다. 센서 노드의 경우 수집된 데이터를 서버로 전송하는 것이 목적이기 때문에 서버에서 제공하는 공개키를 이용해서 암호화를 한 후에 전송하면 된다. 암호화된 데이터가 서버가 아닌 공격자에 의해서 데이터 복사가 되더라도 데이터를 복호화를 할 수 있는 키는 서버만 가지고 있기 때문에 문제가 없게 된다. 또한 데이터의 암호화 정책도 다른 방식으로 접근 할 필요가 있다. 센서 노드에서는 수집된 데이터를 암호화를 하는 알고리즘만 필요하지 굳이 복호화를 하는 알고리즘은 필요치 않게 된다. 서버와의 통신이 종속적이고 데이터의 흐름이 단방향이기 때문에 센서 노드에서는 복호화의 작업이 필요가 없게 된다. 다시 말하자면 센서 노드에서는 암호화에 필요한 공개키만 소유하고 있으면 되고, 센서 노드 자체에 복호화를 위한 키 혹은 인증은 필요 없게 된다.



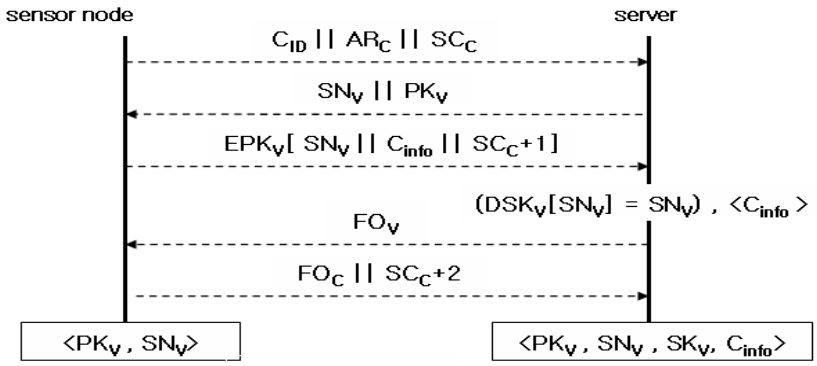

[그림 3-3] 센서 노드와 서버와의 관계

[그림 3-3]과 같은 구조를 가진 환경에서 해결해야 할 사항은 센서 노드가 사용해야 할 공개키가 인가가 된 서버의 공개키를 소유하는 인증이 필요하다. 또한 서버에서는 인가되지 않은 센서의 공격으로부터 네트워크를 보호하기 위해서는 네트워크 내의 센서에 대한 관리가 필요하게 된다. 이러한 점들을 해결 한다면 유비쿼터스 센서 네트워크에서 센서 네트워크의 구조적인 특징을 이용한 공개키 기반의 인증 프로토콜 구현이 가능할 것이다. 본 장에서는 유비쿼터스 센서 네트워크를 위한 경량 보안 프레임워크 중에서 공개키를 이용한 보안 프로토콜에 대하여 설명한다. 공개키를 이용한 보안 프로토콜은 센서 노드와 중앙 처리 서버 간의 안전한 인증을 통한 공개키의 분배를 하는 공개키를 이용한 인증 프로토콜과 공개키 교체 및 관리 프로토콜로 이루어져 있다. 3.2.2 절에서는 공개키를 이용한 인증 프로토콜에 대하여 3.2.3 절에서는 공개키를 이용한 키 교체 및 관리 프로토콜에 대하여 설명한다.

### 3.2.2 공개키를 이용한 인증 프로토콜

[그림 3-3]과 같은 환경에서 서버의 역할은 센서 노드에서 요구하는 공개키를 생성 관리를 하며 센서 네트워크 내의 센서들에 대한 관리 및 네트워크를 감시하는 역할 등 대부분의 보안 정책과 관련된 작업을 한다. 센서 노드에서는 단지 인가된 서버에서 제공하는 공개키를 전송 받고 수집 된 데이터를 암호화 하여 전송하는 작업을 담당하게 된다. 센서 노드가 인가된 서버에서 제공하는 공개키를 전송 받기 위해서 공개키를 이용한 인증 프로토콜을 사용하여 서버와 통신을 하며 이 과정에서 센서 노드는 인가된 서버로부터 공개키를 전달 받고 서버는 새로운 센서 노드에 대한 정보와 해당 센서 노드의 공개키 관리를 하기 시작한다. 서버에서는 공개키를 사용하려는 센서의 정보를 미리 저장하고 있으며 센서 노드별로 다른 공개키를 제공하여, 서버에서는 해당 센서 노드의 센서 정보, 센서 ID, 시큐어 넘버, 서버에서 제공한 해당 센서의 공개키 등의 확인으로 센서를 구별 및 인증하게 된다. [그림

3-4]에서는 센서 노드와 중앙 처리 서버에서 공개키를 이용한 인증 프로토콜의 구조를 나타내었다.




[그림 3-4] 공개키를 이용한 인증 프로토콜

[그림 3-4]에서 사용 하는 기호의 의미는 다음과 같다.

$C_{ID}$ = 센서 ID

$AR_r$ = 센서 노드에서 보내는 인증 요청 신호

|| = 이어 붙여 쓰기

SCc = 센서 노드에서 생성한 시큐어 카운터

SNv = 서버에서 설정한 시큐어 넘버

PKv = 서버에서 생성한 공개키

SKv = 서버에서 생성한 개인키

EPKv[ ] - 서버의 공개키를 이용하여 암호화

DSKv[ ] = 서버의 개인키를 이용하여 복호화

$C_{info}$ = 센서의 고유 정보

FOc = 센서 노드에서 보내는 종료 신호

FOv = 서버에서 보내는 종료 확인 신호

< > = 각 객체에서 저장해야 하는 값

서버와 센서 노드 간의 키 교체 관리 순서는 다음과 같다.

(1) 센서 노드 → 서버 : $C_{ID}$ || $AR_r$ || SCc

(2) 서버 → 센서 노드 : SNv || PKv

(3) 센서 노드 → 서버 : EPKv[ SNv || $C_{info}$ || SCc+1 ]

(4) 서버 : DSKv[ SNv ] = SNv , < $C_{info}$ >

서버 → 센서 노드 : FOv

(5) 센서 노드 → 서버 : FOc || SCc+2

[그림 3-4]의 공개키를 이용한 인증 프로토콜은 다음과 같은 순서로 동작이 된다. 센서 노드에서 센서 노드가 동작을 시작하게 되면 먼저 서버와 인증 작업을 하기 위해서 신호를 보내게 된다($C_{ID}$ || $AR_r$ || SCc). 센서 노드는 서버와의 인증 작업이 끝나기 전까지 어떠한 작업도 하지 않는다.

두 번째 단계에서 서버에서는 해당 센서 노드의 정보를 받은 후에 공개키와 개인키를 생성 하고 시큐어 넘버를 생성해서 센서 노드로 보내게 된다(SNv || PKv). 서버에서는 해당 센서 정보와 연관 있는 개인키, 공개키, 시큐어 넘버를 보관한다. 공격자에 의해서 인증 요청이 오더라도 공개키와 시큐어 넘버는 공개가 되어도 상관이 없으며 다른 센서 노드와는 무관하기 때문에 안전하다고 볼 수 있다. 또한 서버에서는 인증 요청을 하는 센서의 정보를 미리 가지고 있기 때문에 저장되지 않은 센서 노드로부터 요청이 오면 공격자의 공격을 확인 할 수 있게 된다.

세 번째 단계에서 센서 노드에서는 받은 공개키를 이용해서 서버로부터 받은 시큐어 넘버와 센서 노드 자신의 정보를 암호화 하여 전송하게 된다(EPKv[ SNv || $C_{info}$ || SCc+1 ]). 공개키를 이용하여 암호화하였기 때문에 공개키를 전송 한 서버 이외는 복호화가 불가능하기 때문에 공격자가 패킷을 복사를 하더라도 내용을 확인 하지 못하게 된다.

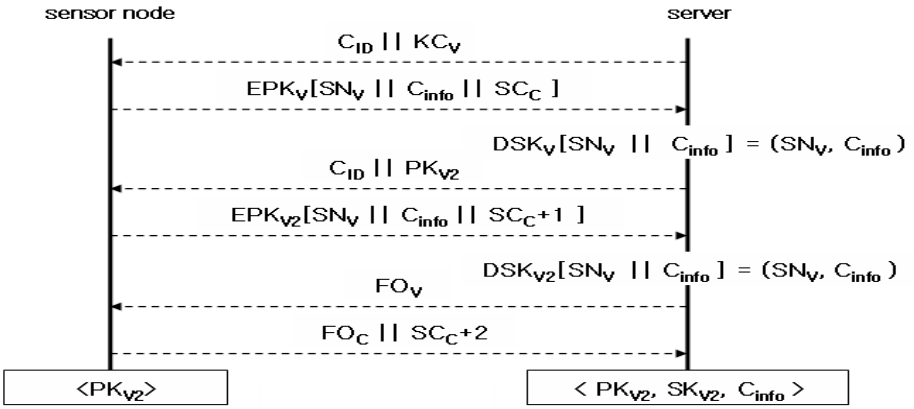
네 번째 단계에서 서버에서는 전송 받은 데이터를 서버에서 보관 중인 개인키로 복호화를 한다. 복호화를 해서 나온 데이터와 자신이 가지고 있는 시큐어 넘버를

비교하여 현재 인증을 받으려는 센서 노드로부터 왔는지를 확인을 한다(DSKv[ SNv ] = SNv). 확인이 끝나고 나서 센서 노드로 인증 종료 신호를 보낸다(FOv).

다섯 번째 단계에서 센서 노드에서도 서버로부터 종료 신호를 받은 후 종료 서버로 종료 확인 신호를 보낸다(FOc || SCc+2). 이러한 작업이 종료가 되기 전 다른 상황이 발생하면 진행 중인 과정은 취소가 된다. 모든 작업이 끝나고 종료 신호까지 전송을 하면 센서 노드에서는 서버의 공개키와 시큐어 넘버를 저장한다. 서버에서는 개인키, 공개키, 시큐어 넘버, 그리고 해당 센서의 정보를 저장한다.

### 3.2.3 공개키를 이용한 키 교체 및 관리 프로토콜

공개키 기반의 암호 정책을 사용하더라도 키의 크기와 키 사용 기간에 따라 공개키를 교체하는 것이 네트워크를 안정적으로 유지하는데 필수적이다. 일반적인 네트워크에서 공개키 기반의 암호 정책은 신뢰가 가능한 인증 서버가 존재 하고 이를 이용한 정책을 사용하기 때문에 공개키 교체 및 분배가 쉬운 편이다.




[그림 3-5] 공개키를 이용한 서버와 센서 노드간의 키 교체 프로토콜

하지만 [그림 3-3]과 같은 환경에서 양쪽의 관계가 종속 적인 상태에서 공개키

분배는 인증 보다 더 중요한 부분을 차지하고 있다. 공개키가 신뢰된 서버로부터 온 공개키인지 확인이 필요하고 또한 센서 노드가 서버에서 분배한 키를 안전하게 보관했는지도 인증만큼 중요한 문제이다. 본 절에서는 유비쿼터스 센서 네트워크의 하드웨어적인 특징과 환경을 고려한 공개키 교체 및 관리 알고리즘을 제안한다.

[그림 3-5]에서 사용 하는 기호의 의미는 다음과 같다.

$C_{ID}$ = 센서 ID
KCv = 서버에서 보내는 공개키 변경 신호
|| = 이어 붙여 쓰기
SNv = 서버에서 설정한 시큐어 넘버
PKv , SKv = 서버에서 생성한 공개키, 개인키
$PKv_2$ , $SKv_2$ = 서버에서 생성한 새로운 공개키, 개인키
EPKv[ ] = 서버의 공개키를 이용하여 암호화
DSKv[ ] = 서버의 개인키를 이용하여 복호화
$C_{info}$ = 센서의 고유 정보
FOc = 센서 노드에서 보내는 종료 신호
FOv = 서버에서 보내는 종료 확인 신호
< > = 각 객체에서 저장해야 하는 값

서버와 센서 노드 간의 키 교체 관리 순서는 다음과 같다.

(1) 서버 → 센서 노드 : $C_{ID}$ || KCv
(2) 센서 노드 → 서버 : EPKv[ SNv || $C_{info}$ || FOc ]
(3) 서버 : DPKv[ SNv || $C_{info}$ ] , ( SNv , $C_{info}$ )
서버 → 센서 노드 : $C_{ID}$ || $PKv_2$
(4) 센서 노드 → 서버 : $EPKv_2$[ SNv || $C_{info}$ || FOc+1 ]

(5) 서버 : $DPK_{v2}[\ SNv \parallel C_{info}\ ]$ , ( SNv , $C_{info}$ )

서버 → 센서 노드 : FOv

(6) 센서 노드 → 서버 : FOc || FOc+2

서버에서는 공개키 교체 주기에 또는 유비쿼터스 센서 네트워크 보안 관리자(uSNSM)의 판단에 의해 공개키 교체 신호를 센서 노드로 전송하게 된다($C_{ID}$ || KCv). 이때 보내는 값으로는 교체를 해야 하는 센서 노드의 값과 센서 노드에서 처리를 해야 하는 교체 명령어가 포함이 되어 있다.

두 번째 단계에서 센서 노드에서는 서버의 교체 신호가 오면 저장해 두었던 시큐어 넘버와 센서 정보를 기존의 공개키로 서버로 전송을 하게 된다($EPK_v[\ SNv \parallel C_{info} \parallel FOc\ ]$). 이때 센서 노드에서는 기존의 공개키를 사용하기 때문에 개인키를 가지고 있는 신뢰된 서버만이 해당 암호를 해독할 수 있게 된다.

세 번째 단계에서 서버에서는 전송된 데이터를 기존의 개인키로 복호화를 하고 시큐어 넘버 값과 센서 정보 값이 기존에 저장되어 있던 값과 일치하는지 검사를 한다($DPK_v[\ SNv \parallel C_{info}\ ]$). 만일 서버에서 키 교체 요청을 하지 않았는데도 해당 값이 전달되었다면 공격자의 공격이라고 판단하게 된다. 또한 공개키 교체 신호를 전송했더라도 응답이 없으면 공격자의 공격이라고 판단하게 된다. 서버에서 데이터 값이 일치를 하면 새로운 공개키를 생성하여 전달하게 된다($C_{ID} \parallel PK_{v2}$). 전송되는 값은 해당 센서의 ID와 공개키이기 때문에 값이 복사가 되더라도 문제가 없게 된다. 센서 노드에서는 공개키 교체 신호를 받고 응답을 했지만 새로운 키를 전송받지 못하면 공격자의 공격으로 판단한다.

네 번째 단계에서 새로운 공개키를 받은 센서 노드에서 새로운 공개키를 이용해서 저장된 시큐어 넘버와 센서 정보를 암호화를 해서 다시 서버에 응답을 한다($EPK_{v2}[\ SNv \parallel C_{info} \parallel FOc+1\ ]$). 공격자의 공격에 의해서 다른 공개키를 이용해서 공격자에게 데이터를 보내더라도 공격자의 공격 여부만 판단하면 신뢰 받은 서버와 센서 노드간의 시큐어 넘버와 센서 정보는 수정이 가능하기 때문에 공격자에게는 의미 없는 정보가 된다.

다섯 번째 단계에서 서버에서는 센서에 분배된 키가 안전하게 전송 되었는지 확인하기 위해서 센서 노드에서 전송한 데이터를 새로운 개인키를 이용해서 복호화를 한 뒤 확인을 한다($DPK_{v2}[ SNv \| C_{info} ]$). 값이 틀리거나 복호화가 불가능 할 경우 공격자의 공격으로 판단하게 된다. 모든 검사가 끝나고 나면 서버에서 센서 노드로 키 교체 알고리즘 종료신호를 보내고(FOv), 여섯 번째 단계에서 센서 노드에서도 서버로 키 교체 완료 신호를 보내게 된다($FOc \| FOc+2$). 이 작업이 끝나면 각각의 센서 노드와 서버에서는 새로운 공개키, 개인키를 저장하게 된다.

기존의 공개키 방식은 암호화를 하는 키를 공개 하는 방식이기 때문에 신뢰가 있는 인증 서버를 이용하여 쉽게 키 관리 및 분배를 하지만 유비쿼터스 센서 네트워크와 같이 종속적인 관계의 양 노드에서 공개키를 이용한 보안 정책을 가질 경우 인증 및 공개키 분배가 어려운 문제로 부각된다. 본 절에서는 이러한 문제를 해결하고자 새로운 공개키 분배 프로토콜을 제안하였다. 본 논문에서 제안된 프로토콜의 경우 시작의 제한과 데이터의 비교 등으로 중간에 공격자의 공격이라고 판단되면 모든 과정은 취소가 되고 중앙 관리 서버에서 처리를 하게 되므로 안전한 공개키 분배라고 할 수 있다.

## 3.3 제안 프로토콜의 안전성 분석

유비쿼터스 센서 네트워크에서 공격자로부터 공격을 받을 수 있는 항목과 유비쿼터스 센서 네트워크 보안 정책이 해결해야 할 사항은 <표 2-1>에 명시를 하였다. <표 2-1>에서의 보안 항목 중 기밀성은 암호화 알고리즘에서 담당을 하게 되며 본 논문에서 제안하는 경량 보안 프로토콜은 인증, 무결성, 가용성 항목의 보안을 해결하였다. 본 절에서는 더블키를 이용한 보안 프로토콜과 공개키 기반의 보안 프로토콜을 사용하고 있을 경우 및 일반적인 센서 네트워크에서 공격자로부터 공격받을 수 있는 공격의 유형과 본 논문에서 제안하는 보안 프로토콜 사용 시 처리 할 수 있는 방안에 대하여 설명하여 본 논문에서 제안하는 더블키를 이용한 보안 프로토

콜의 안전성에 대하여 논의한다. 본 논문에서 제안하는 보안 프로토콜을 사용할 경우 인가되지 않은 사용자로부터 받을 수 있는 공격은 다음과 같이 예상 된다.

- 재전송 공격

  공격자에 의한 재전송 공격은 크게 보안 프로토콜 실행하고 있을 경우와 일반 패킷 전송 시로 구분할 수 있다. 먼저 일반 패킷 전송 시에는 센서 노드에서 생성하는 패킷 안에 본 논문에서 제안하는 보안 정책 중 하나인 시큐어 카운터(SCc) 값이 포함되기 때문에 하나의 패킷을 공격자가 복사를 해서 계속 보내게 될 경우 관리 서버에서 시큐어 카운트(SCc) 값을 대조 할 수 있게 된다. 시큐어 카운트(SCc) 값은 암호화를(EK[data]) 하며 값이 계속적으로 (SCc+N, N=n, n+1, n+2,...) 변하여 전송하기 때문에 수정하기가 쉽지 않다. 따라서 패킷을 복사하여 재전송 공격을 할 수는 있지만 서버에서 바로 확인이 가능하고 또한 본 논문에서 제안하는 감시 정책에 의해서 보안 레벨을 강화 및 센서 노드의 정지 등의 정책이 있기 때문에 안전하다고 볼 수 있다.

  또한 보안 프로토콜 실행 시의 재전송 공격은 시큐어 카운트(SCc)와 시큐어 넘버(SNv) 정책에 의해서 서버에서 확인이 가능하다. 공격자가 인증을 위한 패킷이나 키 교체를 위한 패킷을 복사를 하더라도 서버에서는 인증을 하려는 센서의 정보를 미리 포함하고 있기 때문에 하나의 센서 정보를 이용하여 두 번 이상의 인증 및 키 교체 요구가 올 경우 [그림 5-2]의 서버의 보안 관리자 중에서 보안 레벨 결정 모듈(Security Level Decision module)에서 공격자의 공격이 확인 가능하며, 인가되지 않은 센서가 새로운 패킷을 보내더라도 서버에서 지정한 상태가 되지 않는다면 공격자의 공격 여부를 파악할 수 있게 된다. 재전송 공격은 관리 서버가 그 상황을 즉시 파악 할 수 있기 때문에 감시 정책(4.3절) 그리고 보안 레벨 관리 정책(4.3.1절)에 의해서 즉시 안정화 작업이 가능하다.

- 패킷의 복사, 가로채기, 수정 후 재전송

패킷의 복사, 가로채기, 수정 후 재전송 등과 같은 공격자의 유형은 관리 서버에서 센서 노드로부터 오는 패킷의 간격, 보안 프로토콜 실행 시 진행되는 시간 등을 점검 하고 있기 때문에 발견이 가능하다(4장). 패킷의 복사의 경우 공격자에 의해서 패킷이 복사가 되더라도 중요 항목은 암호화가 되어 있기 때문에 재전송 공격 이외에는 다른 공격을 하기 어렵게 된다. 또한 가로채기를 하여 패킷을 소멸시키게 되면 서버에서 시큐어 카운터 정책(4.1절)과 전송 간격을 측정하고 있기 때문에 확인이 가능하며, 수정 후 재전송의 경우에도 대칭키의 유출되어 해당 센서가 전복이 되지 않는 이상 공격하기 어렵게 되며 만약 하나의 센서가 전복 되었다고 하더라도 다른 센서 노드와 센서 네트워크에 미치는 영향은 적어서 네트워크 전체가 전복 되는 상황은 저지할 수 있게 된다.

- 인가되지 않은 센서 노드의 네트워크 가입

  인가되지 않은 센서 노드가 인가를 받기 위해서는 서버의 인증과정이 필요하다. 하지만 인증은 관리 서버에서 지정한 상황에만(3.1.2절, 3.2.2절) 처리 되므로 인가되지 않은 센서 노드가 인증을 받고 네트워크를 구성할 수는 없게 된다. 기존 센서의 정보를 복사하여 인증을 받고자 하더라도 두 개의 데이터가 중복이 되거나 서버에서 인증 처리를 거부하기 때문에 인가되지 않은 센서 노드의 인증 시도는 즉시 관리 서버에 의해서 발견이 된다.

- 대칭키 전복

  공격자에 의하여 대칭키가 전복되더라도 하나의 센서 노드에 국한된 상황이기 때문에 다른 센서 노드와 센서 네트워크에는 영향이 적게 된다. 또한 키 교체 프로토콜이 주기적으로 또 비주기적으로 이루어지기 때문에(3.1.3절, 3.2.3절) 센서 노드가 전복되기는 쉽지 않으며 하나 이상의 센서 노드가 동시에 전복이 되는 상황은 거의 없게 된다.

- 서비스 거부 공격

  센서 노드의 동작 혹은 관리 서버의 서비스를 방해하기 위한 공격인 서비스 거부 공격의 경우 복사가 된 센서 노드 혹은 라우팅을 하는 센서의 경로 파악이 가능하게 되므로 서비스 거부 공격을 받을 경우 파악이 가능하다. 서비스 거부 공격은 먼저 공격자의 공격을 받고 나서 수동적인 대안만으로 처리가 가능하지만 차후 제안한 보안 정책에 시큐어 라우팅 정책을 포함하여 공격을 당하는 동시에 처리를 할 수 있는 연구가 필요하다.

기존의 보안 정책은 공격자의 공격을 방어하는 수준의 정책이었다면 본 논문에서 제안하는 보안 정책의 경우 공격자의 공격에 대응을 하고 예측을 하여 대비하는 단계까지 설계를 하여 보안 능력을 향상시켰다.

## 3.4 센서 노드 플랫폼과 보안 프로토콜의 관계

본 논문에서 제안하는 유비쿼터스 센서 네트워크를 위한 경량 보안 프레임워크는 인증 및 키 교체 관리 프로토콜과 센서 네트워크 감시 프로토콜로 이루어져 있다. 인증 및 키 교체 관리 프로토콜은 대칭키 기반 알고리즘과 공개키 기반 알고리즘 중에서 한 가지를 선택하여 센서 네트워크 감시 프로토콜과 보안 프레임워크를 구성하도록 되어있다. 현재 유비쿼터스 센서 네트워크 보안에서는 대부분 대칭키 기반의 보안 알고리즘을 사용하고 있다. 대칭키 암호화의 경우 RSA[55] 공개키 암호화 방식에 비하여 소프트웨어인 경우에는 100배의 속도 차이가 나고 하드웨어 인 경우 1000배의 차이가 난다. 또한 56bit의 대칭키를 사용할 경우의 보안 수준과 같은 효과를 얻기 위해서는 RSA 공개키를 기준으로 약 7배가 더 큰 384bit의 공개키가 필요하다. 또한 암호화 및 복호화를 처리하기 위한 모안 모듈의 크기는 공개키 암호와 방식이 대칭키 암호와 방식보다 약 3배 정도 크기 때문에 그 만큼의 저장 공간이 더 필요로 하게 된다. 이러한 하드웨어적인 차이와 암호화 알고리즘의 효율

성의 문제로 현재 유비쿼터스 센서 네트워크의 보안은 대부분이 대칭키 방식을 사용하고 있다. 따라서 현재의 센서 노드의 하드웨어 스펙에서 사용하는 대칭키의 효과를 얻기 위해서 필요한 RSA 공개키를 위한 센서 노드의 하드웨어 스펙은 clock은 약 100배 Program Memory는 3배의 하드웨어 스펙의 발전이 있어야 된다는 결론에 도달 할 수 있다.

최근 들어 공개키 암호화 방식인 타원 곡선 암호(ECC:Elliptic Curve Cryptographic)[56]를 이용하여 센서 네트워크에 적용을 하려는 연구들이 늘어나고 있다[10][36]. 타원곡선(Elliptic Curves)은 약 150년 전부터 수학적으로 광범위한 연구가 있어 왔고, 최근 Andrew Wiles의 Fermat's Last Theorem 증명에서 중요하게 사용되기도 하였다. 타원 곡선 암호 시스템은 비트당 안전도가 타 공개키 시스템보다 효율적이라는 것이 알려졌고, 최근 높은 속도의 구현이 가능하게 되었다. ECC는 유한체 상의 타원 곡선이 유한군을 가지며 그 위에서 이산대수 문제가 구성될 수 있음에 착안되어 제시된 암호 알고리즘이다. Diffie- Hellman 키 분배나 ElGamal암호는 이산대수문제에 근거한 방식이며, 그러한 방식에 대응하는 암호 키 분배를 타원 이산대수 문제 상에서 구성할 수 있다. 이러한 타원 곡선 암호는 1985년에 Koblitz와 Miller에 의해 독립적으로 제안되었다. 타원 이산대수 문제는 키의 크기를 비교적 작게(100~200비트)할 수 있는 실용상의 이점이 있다. 타원 곡선 암호 알고리즘을 사용 할 경우, RSA 공개키의 1024 bit를 기준으로 ECC 공개키는 160 bit만 있으면 같은 보안 수준의 효과를 볼 수 있다. 키의 크기가 줄어든다는 것은 실용적 측면에서 처리 속도나 메모리 사이즈에 영향을 주기 때문에 효율적이라고 할 수 있다. 대칭키 기반의 암호와 정책과 키의 크기를 비교하면 <표 3-1>과 같다.

<표 3-1> 같은 보안 효과를 얻는 키 크기 비교

| 대칭키 암호 | 공개키 암호 | 타원 곡선 암호 |
|---|---|---|
| 80 bit | 1024 bit | 163 bit |
| 128 bit | 3702 bit | 283 bit |
| 256 bit | 15360 bit | 571 bit |

또한 타원 곡선 암호 알고리즘은 RSA 암호 알고리즘에 비해 약 10배 정도 빠르다. 타원 곡선 암호 알고리즘의 특징으로는 기존의 공개키 암호 방식에 비해 단위 비트당 안전도가 높으며 키 크기가 작으며 구현 시 암호화와 서명이 빠르다. 그리고 스마트카드나 휴대 통신기처럼 작은 하드웨어에서 적용하기 쉬우며, 계산량이 작고 저장이 유리하다. 이러한 장점에도 불구하고 타원곡선 알고리즘의 약점과 문제점이 완벽하게 검증받지 못했기 때문에 아직 실용화되지 못하고 있다. 공개키 방식의 타원 곡선 암호를 유비쿼터스 센서 네트워크에서 사용 할 경우 현재의 센서 노드 하드웨어를 기준으로 clock 속도가 10 배가 정도가 되면 현재의 대칭키 수준의 속도와 처리량을 갖출 수 있을 것으로 예상된다. 현재 센서 노드의 하드웨어 스펙을 살펴보면 <표 3-2>와 같다.

<표 3-2> 센서 노드의 하드웨어 스펙

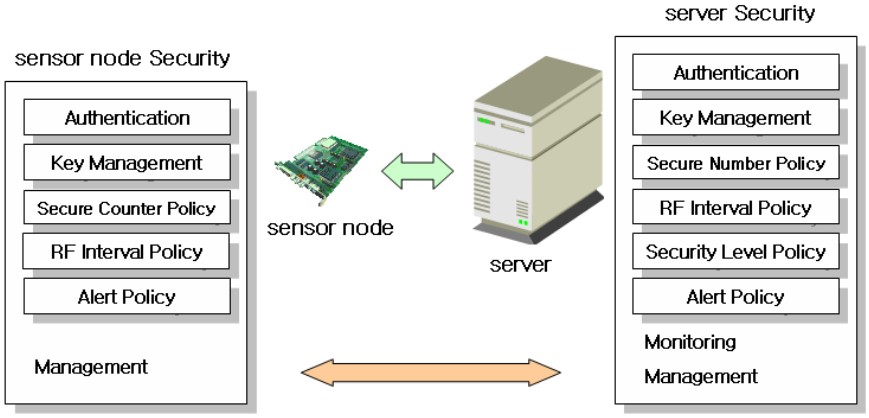
| CPU | Atmel ATmega128L | ARM7TDMI | Texas Instruments MSP430 |
|---|---|---|---|
| Released | 2003 | 2003 | 2004 |
| Clock | 7.37 MHz | 12 MHz | 8 MHz |
| Program Memory | 128 KB | 512 KB | 256 KB |

현재 Atmel의 ATmega128L를 기준으로 할 때 프로세서는 18개월에서 24개월마다 성능이 2배가 되는 무어의 법칙에 따르면 약 8년 뒤인 2011년 이면 지금의 CPU 성능의 10배가 되는 프로세서를 사용하게 될 것이다. 따라서 현재의 Atmel의 ATmega128L을 이용한 센서 네트워크에서 사용하는 대칭키 기반의 보안 정책과 같은 성능을 얻는 공개키 기반의 타원 곡선 암호 알고리즘을 사용하기 위해서는 5년 후인 2011년 이후에는 가능하게 된다. 앞선 3.2절에서 논의 했지만 공개키 방식이 유비쿼터스 센서 네트워크에서는 더 많은 장점을 가지고 있기 때문에 현재 센서 네트워크의 보안 정책의 대부분을 차지하던 대칭키 기반의 보안 정책에서 향후 5년

이내 현재의 대칭키 기반의 센서 보안 정책과 비슷한 수의 공개키 기반의 센서 보안 정책들이 사용될 것으로 예상 된다.

# 제 4 장 유비쿼터스 센서 네트워크 감시 프로토콜

유비쿼터스 센서 네트워크 보안 기술에서 인증, 암호화 정책 등 일부의 보안 정책만으로 보안이 정의 되는 것은 논리적으로 맞지 않고 그 이외 다른 보안 정책들과 같이 유기적인 구조로 이루어져야 하며 이것에 대한 설명이 있어야 할 것이다[그림 4-1]. 본 논문에서는 인증, 키 교체 관리 등의 기본적인 보안 정책뿐만 아니라 보조적인 보안 정책으로 시큐어 카운터 정책과 전송 제한 정책을 이용할 수 있게 하여 보안 관리자가 보안 레벨을 조절 하고 모니터링 할 수 있도록 하였다. 본 장에서는 유비쿼터스 센서 네트워크에서 인증 및 키 교체 관리 프로토콜과 시스템 내에서 유기적으로 동작 할 수 있도록 하는 센서 네트워크 감시 프로토콜에 대하여 설명한다.



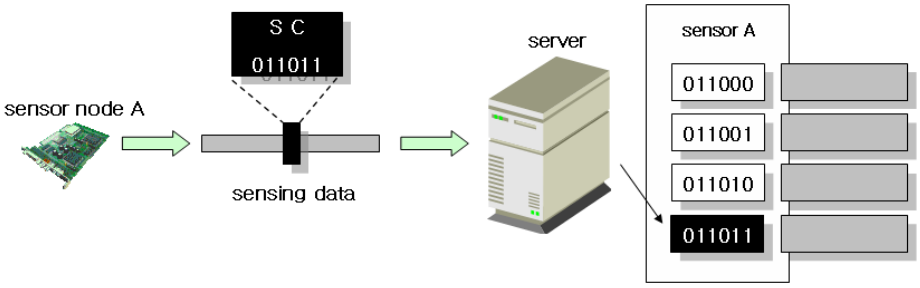

[그림 4-1] 서버와 센서 노드 간의 보안 프레임워크 구조

기존의 센서 네트워크 정책은 수동적인 보안정책으로 공격자의 공격에 능동적으로 대처하는 보안 정책은 아직까지 연구가 미비한 실정이다. 본 논문에서는 모니터링을 통해 얻어진 정보를 토대로 시스템 관리자 혹은 보안 관리자가 결정된 사항에 맞도록 센서 노드의 보안을 강화 하거나, 해당 센서 노드를 네트워크에서 제외시키는 등과 같은 보안 정책을 포함시켰다. 본 4장에서는 앞선 3장에서 설명한 인증 및 키 교환 관리 프로토콜에 포함이 되는 보안 요소 인, 시큐어 카운터 정책, 시큐어

넘버 정책을 설명하고 이 두 가지 정책으로 네트워크의 상태를 모니터링 하여 센서 네트워크 상태를 구분하여 네트워크 상태에 맞는 보안 단계를 바꿀 수 있으며 센서 노드의 보안 단계도 관리 할 수 있는 감시 정책에 대하여 설명한다. 또한 공격자의 공격 상태에 따라서 능동적으로 사용되어 지는 전송 간격 정책, 보안 레벨 관리 정책에 대하여 설명하여 제안하는 경량 보안 프레임워크의 유비쿼터스 센서 네트워크 감시 프로토콜에 대하셔 논의 한다. 이러한 정책은 유비쿼터스 센서 네트워크 보안 정책이 수동적으로만 수립하는 것이 아닌 능동적으로도 사용할 수 있도록 하며 사용자 및 관리자에게 유비쿼터스 센서 네트워크의 상태를 판단하는데 도움이 되어 안전한 네트워크를 유지 할 수 있도록 한다.

## 4.1 시큐어 카운터 정책

기존의 보안 정책에서는 데이터의 신선성(freshness), 무결성 등에 사용할 수 있는 보안 요소로 타임스탬프 프로토콜을 사용한다[20]. 하지만 유비쿼터스 센서 네트워크의 센서 노드의 경우 제한적인 하드웨어 환경 때문에 기존의 타임스탬프를 생성 하거나 타임스탬프 프로토콜을 적용하기에는 문제가 생긴다. 그렇기 때문에 데이터의 신선성, 무결성 등에 사용할 새로운 보안 요소가 필요하다. 본 논문에서 타임스탬프의 보안 요소를 대신할 보안 정책으로 센서 노드에서 생성하는 시큐어 카운터 정책과 서버에서 생성하는 시큐어 넘버 정책을 제안한다. 제안한 두 가지 보안 정책은 센서 네트워크 감시 프로토콜에서 센서 네트워크의 상태 및 센서 노드의 상태를 추정하는 값으로 사용하며 유비쿼터스 센서 네트워크 내의 전송되는 데이터의 신선성과 개체의 인증(authentication), 데이터의 무결성(integrity)에 사용한다. 본 절에서는 유비쿼터스 센서 네트워크를 위한 경량 보안 프레임워크의 보안성을 강화하기 위한 추가적인 보안 정책 중에서 센서 노드에서 생성하여 센서 네트워크 감시 프로토콜에서 처리하는 센서 하드웨어 환경에 맞는 축소된 의미의 타임스탬프 정책인 시큐어 카운터 정책에 대해서 정의하여 설명한다.




[그림 4-2] 센서 노드에서 생성하는 시큐어 카운터 사용 구조

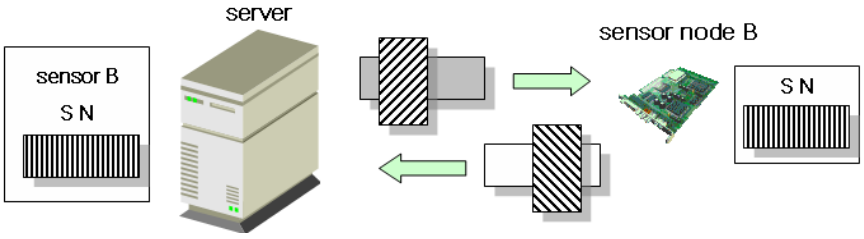
센서 노드에서 주변 환경의 데이터를 수집하여 서버로 전송 할 때 전송하는 패킷에 일정한 크기와 방식의 시큐어 카운터를 포함하여 전송을 하게 된다[그림 4-2]. 주변 환경 데이터를 수집한 값을 전송하는 패킷뿐만 아니라. 센서에서 전송 되는 모든 패킷에는 항상 순차적인 값인 시큐어 카운터가 포함이 되도록 한다. 시큐어 카운터 값은 센서 노드마다 초기 값이 다르기 때문에 센서 네트워크 내에서 사용되는 시큐어 카운터 값은 일정하지 않기 때문에 공격자가 유추해 내기가 불가능하다. 특히 시큐어 카운터 값은 패킷 중에서 암호화로 처리 되는 부분에 포함되기 때문에 두 가지 장점이 있다. 첫째는 센싱한 값이 일정하더라도 시큐어 카운터 값이 변하기 때문에 센서 노드에서 전송하는 패킷의 데이터는 같은 패킷을 전송하는 경우가 없게 된다. 이러한 점은 공격자가 패킷을 중간에서 가로채기를 해서 그 패킷을 재사용을 할 경우 감시 프로토콜에서 쉽게 알 수가 있다. 한 센서에서 보내는 패킷은 같은 값이 없기 때문에 서버에서 공격자의 공격 상황을 쉽게 알 수가 있다. 또한 센싱한 값이 같더라도 패킷의 데이터가 틀리기 때문에 패킷으로 센서에서 사용하도 있는 키를 유추해 내기가 쉽지 않다.

둘째로는 시큐어 카운터의 값이 암호화 되어 있기 때문에 유출된 패킷에서 카운터의 값을 유추해 내지 못하게 되며 이것은 서버에서 패킷을 분석 할 때 어떠한 패킷이 손실 되었으며, 어떠한 센서의 패킷이 공격자에 의해서 재전송되는지를 알 수가 있다. 이러한 두 가지 특징으로 센서 노드에서 전송되는 패킷에 포함되는 시큐어 카운터 값으로 데이터의 신선성과 무결성을 확인할 수 있으며 공격자의 공격 여부까지도 판단 할 수 있는 자료가 된다. 본 논문에서는 센서 노드에서 생성하는 시

큐어 카운터는 2byte의 크기를 갖게 되며 센서 노드가 서버와 인증을 받기 시작하면서 값이 계속해서 서버에 저장되게 된다.

## 4.2 시큐어 넘버 정책

본 절에서는 센서 하드웨어 환경에 맞는 축소된 의미의 타임스탬프 정책인 시큐어 카운터 정책과 시큐어 넘버 정책 중에서 서버에서 생성하는 시큐어 넘버 정책에 대하여 설명한다. 시큐어 넘버 정책은 서버에서 센서로 데이터를 전송할 때 포함되는 데이터로써 주로 센서 노드와 서버 간에 인증을 하거나 키 교체 프로토콜에 의해서 데이터를 전송, 수신할 때 포함이 된다[그림 4-3]. 센서 노드에서 생성되어 포함이 되는 시큐어 카운터 보안 요소와 차이를 두기 위해서 서버에서 생성 되는 데이터는 시큐어 넘버(Secure Number)라는 이름으로 정의되어서 사용되어 진다. 본 논문에서 시큐어 넘버의 값은 4byte의 크기를 가지고 있다. 시큐어 넘버는 센서 노드와 서버의 인증이나 키 교체 알고리즘에서 공격자가 패킷을 가로채기나 변경, 복사의 유무를 서버에서 판단 할 수 있도록 한다.




[그림 4-3] 서버에서 생성하는 시큐어 넘버 사용 구조

센서 노드는 서버로부터 인증을 받기 전까지 아무런 동작을 하지 않게 된다. 서버로부터 인증을 받고 키를 분배 받은 후 정상 작동을 하게 되는데  인증 시 서버로부터 시큐어 넘버를 전송 받게 된다. 이때 생성된 시큐어 넘버는 센서 노드와 서버

양쪽에서 보관을 하게 되는데 이 시큐어 넘버는 키를 교체 할 때 인증 및 교체를 위한 알고리즘에서 사용하게 된다. 공격자가 중간에 복사나 재전송을 하더라도 모든 센서가 각기 다른 시큐어 넘버와 다른 정보를 가지고 있기 때문에 데이터의 무결성을 유지할 뿐 아니라 인증에도 안전성을 유지 할 수 있게 된다. 그리고 보안 관리자에 의해서 교체가 가능하기 때문에 시큐어 넘버의 관리도 가능하게 설계 되어 있다.

## 4.3 서버의 감시 프로토콜

본 논문에서 제안하는 경량 보안 프레임워크는 센서 노드로부터 전달되는 전송 데이터와 인증, 키 교환 시 교환하는 데이터의 분석으로 센서 네트워크 상태를 모니터링 할 수 있도록 설계 되었다. 센서 노드에서 서버로 전송되는 패킷에 포함이 되어 있는 시큐어 카운터와 인증 및 키 교체 관리 시 포함되는 시큐어 넘버를 서버에서 저장 및 분석을 하여 센서 네트워크 감시 및 센서 노드의 상태를 분석한다. 유비쿼터스 센서 네트워크에서 센서 노드는 하드웨어적인 제약이 심하기 때문에 네트워크 상태 분석은 하드웨어의 제약에서 비교적 자유로운 서버에서 하도록 설계하였다. 이러한 점은 센서 노드에는 저전력에 중점을 둔 보안 정책으로 설계를 하며 서버에서는 센서 노드에서 전달되는 데이터 값으로 네트워크의 상태를 파악해서 유비쿼터스 센서 네트워크 전체적으로는 안전한 네트워크를 구성 할 수 있도록 하였다. 유비쿼터스 센서 네트워크 감시 프로토콜은 서버에서 관리하는 시큐어 넘버, 시큐어 카운터를 이용하여 인증 프로토콜 감시, 키 교환 프로토콜 감시, 일반 데이터 감시의 세 가지로 구분할 수 있다.

- 일반 데이터 감시

  시큐어 카운터 정책은 앞선 4.1절에서 설명하였다. 시큐어 카운터는 센서 노드에서 전송 되는 모든 패킷에 포함이 되기 때문에 중앙 처리 서버의 유비쿼

터스 센서 네트워크 보안 관리자(uSNSM)에서는 패킷이 서버로 전송 될 때 항상 데이터의 값을 점검하도록 되어 있다[그림 4-2]. 시큐어 카운터의 값은 센서마다 다른 값을 가지고 있고 해당 센서에서는 연속적인 값이 들어오기 때문에 서버의 보안 모듈인 uSNSM에서 센서 별로 데이터를 감시할 수 있어서 어떠한 센서에 이상이 있는지 네트워크의 상태가 어떤지 쉽게 파악을 할 수 있는 장점이 있다. 또한 공격자에 의하여 인가되지 않은 센서 노드로부터 대량의 데이터를 전송하여 공격도 파악할 수 있다. 서버에는 각 센서 노드마다 데이터의 전송 시간이 정해져 있기 때문에 시큐어 카운터 값으로 확인이 가능하여 대비를 할 수 있도록 설계 하였다. 또한 일정 시간동안 값이 전달되지 않는 것을 파악하여 센서의 이상 여부, 공격자에 의한 센서 노드 전복 등에 대해서도 파악이 가능하다. 센서 노드로부터 데이터가 전송되면 먼저 감시 프로토콜에 의해서 데이터를 감시하도록 설계하였다.

- 인증 프로토콜 감시

  서버와 센서 노드 사이에서 인증 및 키 교환 시에는 공격자에 의해서 데이터의 불법 복사, 재전송, 변조 등의 공격이 쉽게 일어날 수 있다. 본 논문에서 제안하는 인증 프로토콜과 키 교환 프로토콜은 이러한 경우에도 데이터의 기밀성과 무결성을 확인 할 수 있도록 설계 되었다. 인증 프로토콜을 실행 하고 있을 경우 센서 노드에서 보내는 패킷에는 시큐어 카운터 값이 있으며, 인증을 시작하는 순간부터 서버에서 시큐어 넘버가 생성이 되어서 두 개체 간에 데이터 전송 시 확인을 하도록 하였다. 인증은 센서 네트워크의 안전성을 유지 할 수 있는 가장 중요한 부분이기 때문에 인증 프로토콜 감시에는 센서 노드로부터 전송되는 시큐어 카운터 값을 감시 하고, 인증 단계 별로 시큐어 넘버를 감시하며, 인증 프로토콜이 진행되는 전체 시간을 감시하는 세 단계를 통하여 공격자의 공격 여부를 판단하게 된다.

- 키 교환 및 관리 프로토콜 감시

키 교환 및 관리 프로토콜 진행시에도 공격자의 불법적인 복사, 재전송, 변조 등의 공격이 쉽게 일어날 수 있다. 인증 프로토콜 실행 시 감시와 마찬가지로 키 교체 관리 프로토콜 실행 시 감시에도 센서 노드의 시큐어 카운터, 인증 시 교환 되었던 시큐어 넘버, 그리고 키 교체 관리 프로토콜의 진행 시간 감시를 통하여 시간의 지체, 데이터의 변경, 중복 등의 상황을 파악 할 수 있다.

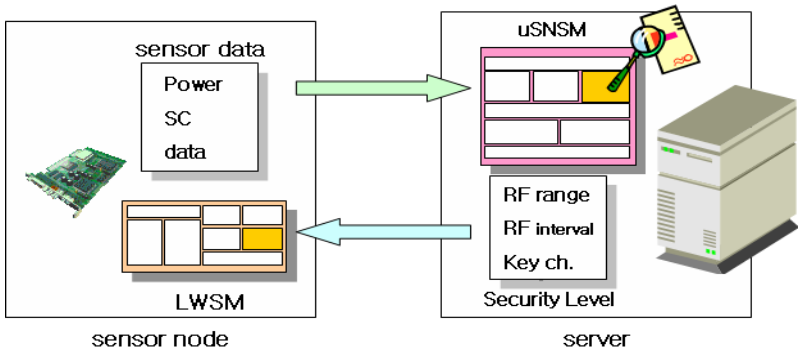
이러한 세 가지 감시 방법에 따라서 센서 노드로부터 데이터가 전송되고 서버에서 수신되는 데이터를 계속 적으로 모니터링 하여 현재의 네트워크 상태를 계속해서 감시 할 수 있으며 센서 노드의 상태도 파악이 가능하다. 이러한 방식은 센서 노드에는 저전력을 유지할 수 있도록 연산량을 줄이고 하드웨어 제약에서 비교적 자유로운 서버에서 각 센서에서 전송되는 데이터의 여러 값만을 분석하여 안전적이고 안전한 네트워크가 유지 될 수 있도록 하는 방식이다. 본 4.3절에서는 이러한 세 가지 감시 프로토콜에 의해서 센서 네트워크의 상태를 유추하여 어떠한 방식으로 네트워크의 보안 레벨을 결정할 지에 대한 정책에 대하여 설명 한다. 또한 네트워크의 상태를 파악하여 어떻게 센서의 상태를 변경할 것인지에 대한 능동적인 보안 정책에 대하여 설명한다.

4.3.1절에서는 감시 정책으로 네트워크의 상태를 모니터링 하여 현재의 네트워크 상태를 파악하는 센서 네트워크 보안 레벨 정책에 대하여 설명한다. 4.3.2절과 4.3.2절에서는 파악이 된 센서 네트워크의 상태에 따라서 센서의 상태를 능동적으로 바꿀 수 있는 보안 레벨에 따른 전송 거리, 간격 정책에 대하여 설명한다.

### 4.3.1 센서 네트워크 보안 레벨 정책

기존의 보안 정책이나 센서 네트워크의 보안 정책에서는 키의 전복 상태를 파악하는 것은 힘들다. 하지만 공격자의 공격 상태를 파악하거나 공격의 공격 시도 상태를 파악하는 것은 가능하다. 기존의 센서 네트워크 보안 정책의 대부분은 주로

공격자의 공격을 대비하는 수동적인 보안 정책을 가지고 있다. 또한 모니터링 결과에 의해서 공격자의 공격 상태가 확인이 되더라도 물리적인 방법으로 센서나 네트워크의 상태를 점검하는 방법이외는 다른 정책이 포함이 되지 않고 있는 실정이다.




[그림 4-4] 센서 노드의 상태에 따른 보안 등급 변경 구조

본 논문에서 제안한 서버의 네트워크 감시 정책에 의해서 인증 및 키 교체 관리 시에 공격자의 공격 여부가 확인 가능하며, 인증이 된 센서 노드 패킷 전송 시에도 항상 서버에서 패킷을 감시하여 현재의 네트워크가 공격자의 공격이 이루어지고 있는지 아닌지를 확인 할 수 있다. 또한 현재의 상태를 보안 레벨 단계로 정의해서 그 상태에 맞도록 센서가 작동하고 네트워크가 동작이 되도록 하는 능동적인 보안 정책을 가지고 있다[그림 4-4].

<표 4-1> 유비쿼터스 센서 네트워크의 보안 상태

| 보안 상태 | 네트워크 특징 | 보안 대처 방법 |
|---|---|---|
| 안정 | • 공격자의 공격이 없는 일반적 상태 | - |
| 공격 시도 | • 공격자가 네트워크나 센서를 전복 시킬 목적으로 패킷의 불법 복사, 변조, 가로채기 등을 시도하는 상태 | • 키 관리, 네트워크 상태 점검<br>• 키 교체 및 센서 상태 점검 |
| 공격 | • 네트워크가 공격자에 의해서 공격 당하는 상태 | • 해당 공격 차단하여 네트워크 상태 안정화 |

4.1절과 4.2절에서 설명한 정책을 이용하여 현재의 네트워크 상태에 따라서 보안 레벨 단계 정의가 가능하다. 보안 레벨은 공격자의 공격이 확실한 상태와 그렇지 않은 상태로 나눌 수가 있다. 그리고 공격이 불확실한 상태는 공격자의 공격이 없는 안전한 상태와 공격자의 공격 시도를 하는 상태로 구분 할 수 있다. 이러한 관점에서 <표 4-1>과 같이 센서 네트워크의 보안 상태를 구분하면 크게 안정 상태, 공격 시도 상태, 공격 상태로 구분이 가능하다.

본 논문에서 제안하는 시큐어 카운터 정책과 시큐어 넘버 정책으로 공격자의 공격 상태를 파악할 수 있다. 또한 공격자의 데이터 변조나 가로채기, 패킷의 재전송 등과 같은 공격 상태로 확인이 가능하다. 하지만 유비쿼터스 센서 네트워크의 특징상 패킷의 손실이 충분히 일어날 수 있는 환경이기 때문에 패킷의 유실이나, 수신 시간의 딜레이 등과 같은 결과만으로는 공격자의 공격 시도 상태인지 단순한 네트워크 상태 불량이나 센서 노드의 작동 오류를 확인하기는 힘들다. 따라서 본 논문에서는 네트워크의 상태와 센서 노드의 상태를 다음 네 가지 단계로 구분을 하고, 보안이 필요한 보안 레벨은 <표 4-2>에서처럼 세 단계로 구분하여 설계하였다.

<표 4-2> 보안 레벨 정책 단계 및 네트워크 상태

| 보안 상태 | 보안 레벨 | 네트워크 상태 |
|---|---|---|
| 안정 | - | - |
| 자연 유실<br>공격 의심 | 주의 | • 네트워크 상태 불량으로 인한 패킷 유실 혹은 공격자의 공격으로 의심 하는 상태 |
| 공격 시도 | 경고 | • 공격자가 네트워크나 센서를 전복 시킬 목적으로 패킷의 불법 복사, 변조, 가로채기 등을 시도하는 상태 |
| 공격 | 위험 | • 네트워크가 공격자에 의해서 공격당하는 상태 |

<표 4-2>에서 구분하는 세 가지 단계의 보안 레벨은 <표 4-3>과 같이 보안 레벨에 따라서 서버에서 센서 노드의 행동을 제어할 수 있다. 센서 노드 자체적으로는 보안 레벨을 변경 할 수 있는 모듈은 없기 때문에 공격자에 의해서 변경되는 경

우는 없다. 센서 노드의 상태 변화는 서버의 보안 레벨 변경 신호에 의해서만 가능하며 감시 프로토콜에 의하여 서버에서 센서 노드로 보내는 패킷 중에서 Data type과 SL항목에 의하여 센서 노드의 상태가 변경하게 된다.

<표 4-3> 보안 상태에 따른 센서 노드 상태 변경

| 보안 레벨 | 센서 노드 상태 변경 내용 |
|---|---|
| 주 의 | 전송 거리 기본, 간격 변경 |
| 경 고 | 전송 거리 최소, 전송 간격 최소, 지정 시간 휴식 |
| 위 험 | 센서 강제 종료 및 센서 지정 시간 휴식 |

본 논문에서는 <표 4-3>의 서버에서 보안 레벨에 따라서 센서 노드의 상태를 변경 할 수 있는 항목은 <표 4-4>과 같이 정의 하였다.

<표 4-4> 서버에서 조정 가능한 센서 노드 항목

| 항목 | 센서 노드가 처리하는 행동 |
|---|---|
| 센서 노드 전송 거리 | • 센서 별 기본 전송 거리<br>• 전송 거리 최소 |
| 센서 노드 전송 간격 | • 센서 별 기본 전송 간격<br>• 최소 전송 간격<br>• 최대 전송 간격 |
| 센서 일정 기간 휴식 | • 서버로부터 신호 올 때까지 휴식 |
| 센서 강제 종료 | • 센서 강제 종료 |

<표 4-4>에서 센서 노드의 전송 거리, 전송 간격 및 강제 종료, 일정 기간 휴식에 관한 내용은 4.3.2절과 4.3.3절에서 설명한다.

### 4.3.2 보안 레벨에 따른 전송 거리 정책

유비쿼터스 센서 네트워크에서 센서 노드의 데이터 전송 거리에 관한 분야는 주로 센서 노드의 저전력과 연관이 깊기 때문에 대부분의 연구에서 보안 정책 관리하고는 연계를 시키지 않는 경향이 있다. 하지만 본 논문에서는 센서 노드의 데이터 전송 거리 제한의 권한과 정책을 유비쿼터스 센서 네트워크 보안 관리자에 포함시켜서 보안에 적극 활용을 하도록 설계 하였다. 센서의 전송 거리를 보안과 연관시키면 다음과 같은 장점이 있다. 첫째, 전송 거리가 최소화 되면 외부에서 무선 패킷을 복사하거나 가로채기 등의 불법적인 공격을 하기가 힘들어진다. 둘째, 전송 거리를 최소화 하여 전력의 출력을 줄여서 저전력에 도움이 된다. 셋째, 브로드캐스팅을 하는 센서 노드의 특징상 최단 라우팅 경로에 있는 센서로만 전송이 가능하기 때문에 네트워크 내의 다른 센서의 보안 정책에 도움이 된다.

<표 4-5> 네트워크 공격 상태에 따른 전송 거리 변경

| 보안 레벨 | 네트워크 상태 | 센서 전송 거리 상태 |
|---|---|---|
| 주 의 | • 공격자의 공격을 의심하는 상태 | • 기본 전송 거리 |
| 경 고 | • 공격자가 네트워크나 센서를 전복시킬 목적으로 패킷의 불법 복사, 변조, 가로채기 등을 시도하는 상태 | • 전송 거리 최소로 설정하여 패킷의 불법 복사, 가로채기 시도에 방어 |
| 위 험 | • 네트워크가 공격자에 의해서 공격당하는 상태 | • 전송 거리 최소로 설정하여 공격자의 불법 적인 공격에 방어 |

본 논문에서 제안한 전송 거리에 대한 정책은 두 가지 상황에서 사용 하도록 하였다. 첫째, 센서 노드가 모든 작업을 중단하고 서버로부터 인증을 받기 위해서 대기를 할 때, 즉 센서 노드가 처음 동작을 하면 자동으로 최단 거리 전송을 하도록 하여 인증 프로토콜에서 사용 하도록 하였다. 둘째, 시스템 설계자가 유비쿼터스 센서 네트워크의 설계를 할 때 센서 노드의 위치나 중요도에 따라서 센서의 데이터 전송 거리를 지정 하여 기본적인 전송 거리를 갖게 되며 보안 레벨에 따라서 서버로부터 전송 거리 변경 신호에 따라서 최소 거리 혹은 기본 거리로 변경 할 수 있

도록 하였다. 어떤 보안 레벨 상황에서 센서의 전송 거리, 전송 간격을 어떻게 변경해야 네트워크를 안정성 있게 그리고 저전력을 유지 할 수 있을까 하는 상관관계는 네트워크의 구성 상태에 따라 다를 수 있다. 본 논문에서는 <표 4-2>에서 보안 레벨에 따른 공격자의 공격 유형을 구분한 것을 기준으로 다음과 같이 대비 할 수 있다.

보안 레벨이 주의 상태 일 경우에는 네트워크의 상태 이상으로 인한 상황인지 공격자의 공격으로 인한 상황인지 구분하기 힘들기 때문에 센서의 전송 거리를 기본 상태를 유지하도록 한다. 보안 레벨이 '경고' 상태 일 경우는 현재 네트워크의 상태가 네트워크의 상태이상으로 인한 상황이 아니라 의도적인 상태이기 때문에 공격자의 패킷의 불법 복사, 가로채기 등을 공격적인 상태를 방어하기 위하여 전송거리를 최소화 하여 센서 노드의 패킷을 방어 하도록 한다. 보안 레벨이 '위험' 상태인 경우는 공격자가 인증된 패킷을 계속해서 재전송을 시도하거나 데이터의 변조 등의 공격을 하고 있는 경우이기 때문에 불법적인패킷들이 네트워크에서 많아져서 정상적인 네트워크의 상태를 방해하기 때문에 전송 거리를 최소화 하거나 문제가 되는 센서 노드는 강제 종료를 하도록 센서의 상태를 변경 시킨다. 이러한 상태를 식으로 나타내면 (식 4-1)과 같다.

$$SN_{RF} = DL - NS \qquad \text{(식 4-1)}$$

여기서 기호 $SN_{RF}$는 센서 노드가 패킷을 전송하는 거리를 나타낸다. 값은 2, 1, 0의 값을 갖게 되며 2일 경우 설계자가 설정한 기본 전송 거리로 패킷을 전송하게 되며 값이 1일 경우 전송 거리를 최대한 줄여서 전송하게 된다. 값이 0이 될 경우에는 전송을 못하게 된다. DL은 보안 설계자가 정한 보안 레벨의 값인데 본 논문에서는 보안 단계를 세 단계로 잡았기 때문에 기본 값은 2가 된다. NS는 네트워크의 상태 값으로 주의 상태 일 경우 값은 0, 경고 상태 일 경우 1, 네트워크의 상태가 위험일 경우 2의 값을 갖는다. 따라서 보안 레벨에 따라서 네트워크의 상태가 정해지게 되면 센서가 전송해야 할 거리가 (식 4-1)에 의해 정해지게 된다.

### 4.3.3 보안 레벨에 따른 전송 간격 정책

센서 노드의 전송 간격은 센서의 저전력과 깊은 관계가 있다. 대부분의 센서 노드 저전력 정책에서 에너지에 따른 전송 간격 변경을 사용하고 있다. 본 논문에서는 저전력 정책뿐만 아니라 보안 정책에도 센서 노드의 전송 간격 변경을 적극 활용하여 공격자의 공격 상태에 따라서 네트워크를 보호 할 수 있도록 제안한다. 4.3.2절에서 센서 노드의 전송 거리 변경으로 공격자의 공격에 대비 할 수 있도록 하였다. 센서 노드의 전송 간격 변경 정책은 공격자의 공격이나 불법적인 상황에 대비할 수 있는 정책으로 사용하기는 무리가 있지만 전송 거리 정책의 보조 정책으로 사용한다면 현재의 네트워크의 공격 상태 여부를 파악할 수 있으며, 센서 네트워크에서 문제가 되는 센서 노드나 혹은 불법적으로 활동하는 노드를 제외시킬 수가 있다.

<표 4-2>의 보안 레벨 상태가 '경고' 일 경우는 서버의 감시 프로토콜에 의해서 공격자의 공격으로 판단 가능한 상태이기 때문에 실제 센서 노드의 전송 간격 변경만으로는 공격자의 공격을 방어 할 수 없다. 하지만 예를 들어 하나의 센서 노드에서 일정한 값의 센싱 데이터가 들어오다가 그 값이 갑자기 커지거나 작아질 경우 감시 프로토콜 알고리즘에 의해서 보안의 상태가 경고 상태가 된다. 실제 공격자가 불법 복사 및 변조를 통하여 값을 변경하여 불법 전송을 하거나 실제 센싱한 데이터의 값이 갑자기 변화가 생길 수 도 있지만 이것을 파악하기는 쉽지 않다. 이럴 경우 보조적으로 전송 간격 변경 정책을 사용하여 센서 노드의 상태를 파악할 수 있다. 또한 센서 노드의 이상이나 불법적인 패킷 공격으로 특정 센서 노드에 문제가 생길 경우 센서를 강제적으로 혹은 일정 기간 휴식 상태로 처리 하도록 하기 위해서 전송 간격 정책을 사용한다. 이러한 전송 간격 정책을 적극적으로 사용하여 수동적인 보안 정책을 가지는 것이 아니라 공격에 적극적으로 방어를 할 수 있는 보안 레벨 정책을 정립 할 수 있다.

센서 노드가 온도, 습도, 압력 등의 어떠한 센싱 데이터 전송을 담당하고 있는지, 또한 센서의 목적에 따라서 센서 노드의 전송 간격은 다르게 설계되며 이것은 시스템 설계자가 결정하게 된다. 센서는 시스템 설계자에 의해서 기본 전송 간격, 최소

전송 간격, 최대 전송 간격, 지정 시간 전송 휴식, 강제 종료의 다섯 가지 상태가 포함이 된다. 전송 간격은 (식 4-2)와 같이 표현 할 수 있다.

$$SN_{IT} = SN_{BIT} \times NS_v \quad \text{(식 4-2)}$$

여기에서 기호 $SN_{IT}$는 센서 노드가 패킷을 전송하는 간격을 나타내며, $SN_{BIT}$는 센서 노드에 저장되어 있는 기본 전송 간격을 나타내며, $NS_v$는 네트워크 보안 상태 값을 나타낸다. 센서 노드의 전송 간격은 각 센서마다 다른 기본 정수 값을 갖게 되며 네트워크 설계자가 센서 노드의 위치와 센싱 목적에 따라서 정해가 된다. $NS_v$는 네트워크 상태에 따라서 '주의' 상태 일 경우 값은 2, 0.5가 되며, '경고' 상태 일 경우 0.5, 10 × n (n=1,2,3...) 값을 갖게 된다. 그리고 '위험' 상태 일 경우에는 0, 10 × n (n=1,2,3...) 값을 갖게 된다. (식 4-2)에서 서버의 보안 관리자의 네트워크 상태 판단에 의해서 전송 간격은 다음 설명과 같이 크게 네 가지의 간격으로 구분 할 수 있다.

- 기본 간격 변경 (네트워크 주의 상태)

센서 노드의 기본 전송 간격($SN_{BIT}$)은 시스템 설계자에 의해서 센서 별로 기본 전송 간격 시간이 정해진다. 각 센서는 데이터 전송 간격을 5초에서 1분 이상 고유한 기본 전송 간격을 보유하게 된다. 보안 관리자에 의해서 특정 센서 노드에 의해서 네트워크 상태가 '주의' 상태($NS_v$ = 0.5, 2)로 변경이 되면 해당 센서 노드의 종류와 초기 설정에 따라서 간격을 변경 하게 된다. 센서 네트워크의 상태가 보안 관리자에 의해서 '주의' 상태가 된다고 해서 모든 센서의 전송 간격이 변하는 것은 아니며 해당 센서의 간격만 변경하게 되며 시스템 설계자의 초기 설정에 의해서 $NS_v$ 값이 0.5나 2의 값을 갖게 된다. 따라서 $SN_{IT}$의 값은 $SN_{BIT}$의 값에서 전송 간격이 2배로 늘어나거나 2배로 줄어들게 된다.

- 전송 간격 최소 변경 (네트워크 경고 상태)

보안 관리자에 의해서 네트워크 '경고' 상태가 되면 전송 간격을 센서노드에 저장된 최소 전송 간격으로 변경하게 된다. (식 4-2)에서 네트워크 상태가 주의 상태($NS_v$ = 0.5, 10 × n(n=1,2,3...))로 변경이 되면 센서 노드의 전송 간격은 반으로 줄어들던지 점차적으로 늘어나게 된다. 네트워크가 '경고' 상태가 되면 공격자의 공격 시도가 이루어지고 있다고 판단이 되어지며 정확한 공격 상태를 파악할 때까지 해당 센서의 최소 간격, 혹은 이미 최소 상태로 전송하고 있는 상태라면 지정 시간 동안 전송을 금지 시키도록 한다.

• 지정 시간 휴식 (네트워크 경고/위험 상태)

보안 관리자가 네트워크의 상태를 '경고' 상태로 판단하게 되면 네트워크 내에서 공격자의 공격이 예상 되고 있는 상태이기 때문에 해당 센서 노드를 일정 기간 데이터 전송을 못하도록 한다. $NS_v$의 값이 10 × n(n=1,2,3...) 중 하나의 값을 갖게 되는데 그 시간만큼 일정 시간 동안 패킷 전송을 금지 시킬 수 있게 된다. 센서 노드가 지정 시간 휴식이 되었다가 다시 전송 간격을 기본 전송 간격으로 유지 하더라도 센서 노드가 계속 해서 지정 시간 휴식을 해야 할 경우는 강제 종료 상태로 넘어가서 네트워크 관리자의 점검을 하도록 한다.

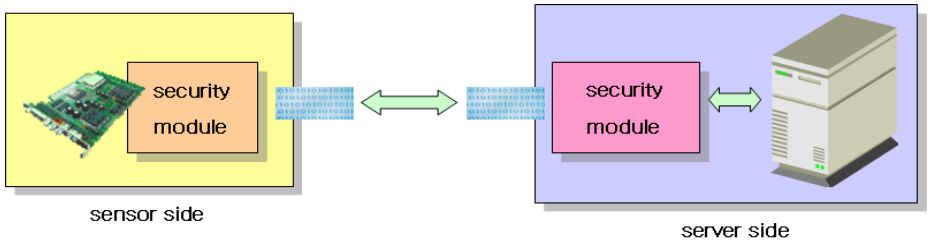
• 강제 종료 (네트워크 위험 상태)

보안 관리자가 네트워크의 상태를 '위험' 상태로 판단하게 되면 공격자의 공격이 진행 되고 있는 상태이기 때문에 그 공격상태를 파악할 수 있다. 제안하는 보안 프레임워크에서는 재전송 공격, 인가되지 않은 데이터 전송, 데이터 변경의 공격자의 공격을 파악 할 수 있기 때문에 어떤 센서가 공격의 목표가 되고, 문제가 발생한지 파악할 수 있다. 공격의 유형에 따라서 전체 네트워크가 전복되지 않도록 하기 위해서 강제 종료를 시킬 수 있도록 한다. $NS_v$의 값이 0이 될 경우는 센서가 강제 종료가 되며 강제 종료가 된 센서 노드는 네트워크 관리자가 센서를 점검 및 교체를 할 때까지 그 동안 사용되었던 해당 센서의 키를 네트워크 내에서 사용하지 못하도록 한다.

# 제 5 장 경량 보안 프레임워크 구현

## 5.1 경량 보안 프레임워크 구조

앞선 [그림 2-1]의 센서 네트워크의 환경을 유비쿼터스 센서 네트워크 보안 정책을 위한 관점에서 살펴보면 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 5-1]에서 서버 측에 어디에 위치를 해도 상관이 없다. [그림 5-1]은 [그림 2-1]의 유비쿼터스 센서 네트워크의 구조를 센서 노드와 서버의 양 객체간의 관계로 간략화 시킨 구조이다.




[그림 5-1] 보안의 객체 개념으로 간략화 된 센서 네트워크의 구조

본 논문에서 제안하는 경량 보안 프레임워크는 인증 및 키 교체 관리 프로토콜과 감시 프로토콜로 이루어져 있다. 두 가지 프로토콜은 [그림 5-1]의 서버 측 보안 모

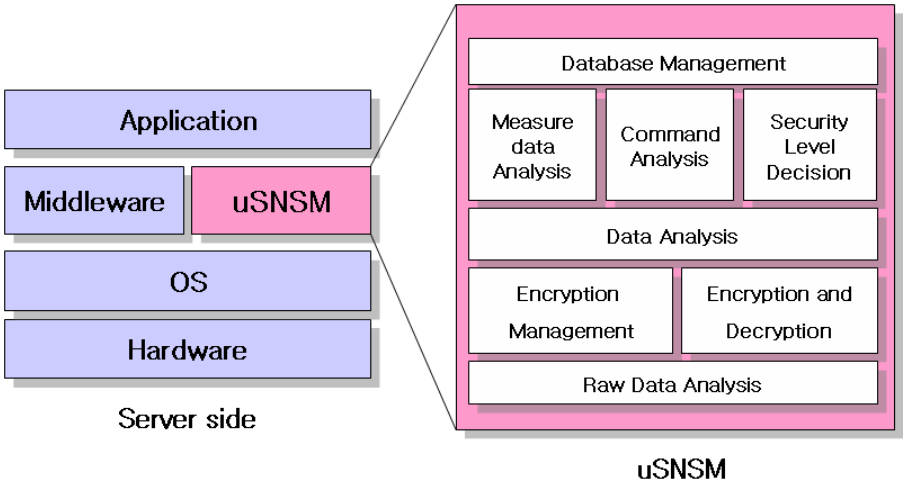
듈과 센서 노드의 보안 모듈 위에서 동작하도록 설계 하였다. 본 절에서는 제안하는 경량 보안 프레임워크를 서버와 센서 노드로 구분하여 제안한 경량 보안 프레임워크를 설계하였다. 5.1.1절에서는 서버의 보안 관리자에 대하여 설명하고 5.1.2절에서는 센서 노드에 포함되는 보안 관리자 구조에 대하여 설명한다.

### 5.1.1 서버의 보안 관리자 구조

본 5.1.1절에서는 유비쿼터스 센서 네트워크를 위한 경량 보안 프레임워크 중에서 서버의 보안 구조에 대해서 설명한다. 본 논문에서는 서버 측의 보안 모듈은 유비쿼터스 센서 네트워크 보안 관리자(Ubiquitous Sensor Network Security Manager)라는 명칭으로 정의해서 사용하며, 센서 측의 보안 모듈은 경량 보안 프로토콜(Light Weight Security Protocol)이라는 명칭으로 정의해서 사용한다. [그림 5-2]의 유비쿼터스 센서 네트워크 보안 관리자(uSNSM)에 대한 구조 및 각각의 모듈에 대해서 설명하면 다음과 같다.

- 로우 데이터 분석 모듈 (Raw 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 않은 부분이 있기 때문에 암호화가 되지 않은 부분에서 필요한 데이터를 추출해서 암호화 된 데이터 부분을 어떠한 알고리즘으로 복호화를 할지, 어떤 공개키나 대칭키를 사용할지를 결정해서 데이터를 처리한다. 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.




[그림 5-2] 유비쿼터스 센서 네트워크 보안 관리자(uSNSM) 구조

- 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 알고리즘 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 시큐어 카운터와 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.

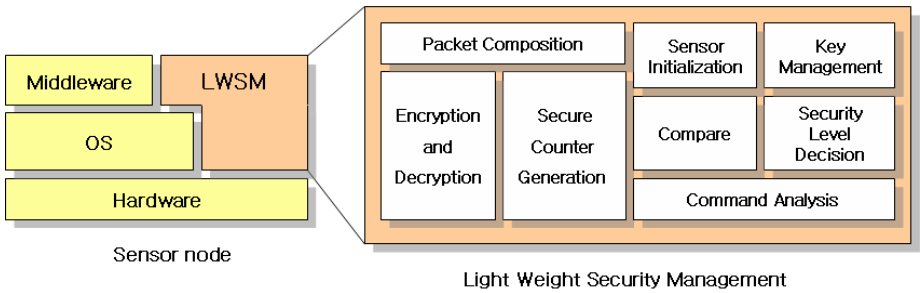
- 데이터 저장 관리 모듈 (Database Management module)

  데이터 저장 관리 모듈은 센서 노드로부터 온 데이터를 저장하거나 다른 응용 프로그램에서 사용할 수 있도록 처리 하는 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.

[그림 5-2]의 모듈 중에서 로우 데이터 분석 모듈 (Raw 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.

### 5.1.2 센서 노드의 경량 보안 관리자 구조

유비쿼터스 센서 네트워크를 위한 경량 보안 프레임워크 구조에서 센서 노드에 포함이 되는 경량 보안 관리자의 구조는 [그림 5-3]과 같다.




[그림 5-3] 센서 노드의 경량 보안 관리자 구조

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. 패킷 구성에 관한 내용은 7장에서 설명하도록 한다. 패킷의 구성은 센서 노드의 운영체제에 종속적이기 때문에 다른 운영체제를 가진 센서 노드들은 다른 패킷으로 구성될 수밖에 없지

만 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 센서의 하드웨어 능력에 따라 복호화를 하도록 처리하는 모듈이다. 실제 패킷 구성 모듈에서 전송될 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 발전하였을 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 처리할 수 있는 모듈이며 센서 노드의 전원 이상이나 센서 노드의 네트워크 내에 탈퇴 여부를 결정할 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.

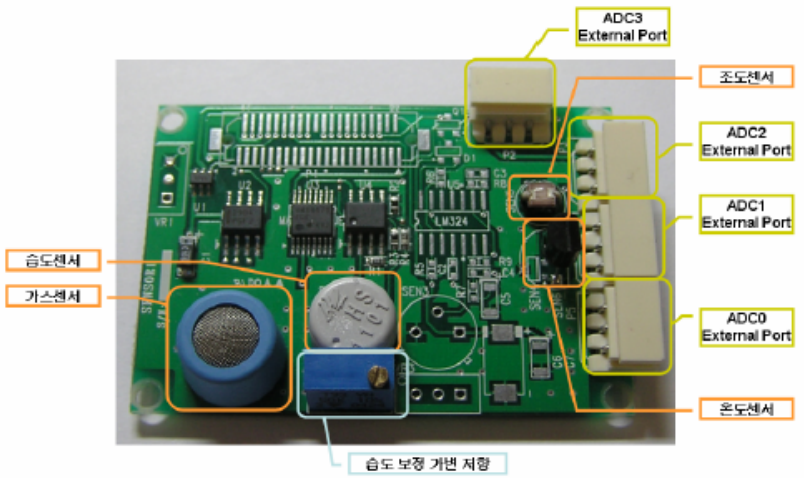
경량 보안 관리자는 센서 노드의 제한적인 하드웨어 특성을 고려해서 설계되었으며 운영체제나 전송 방식에 독립적으로 동작하도록 설계하였다. 다른 보안 정책에서 사용하는 암호화 정책, 암호화 알고리즘을 이용할 수 있도록 할 수 있으며, 각각의 다른 운영체제에서 지원하는 데이터 패킷을 이용할 수 있도록 하여 확장성을 고려하였다.

## 5.2 경량 보안 프레임워크 구현

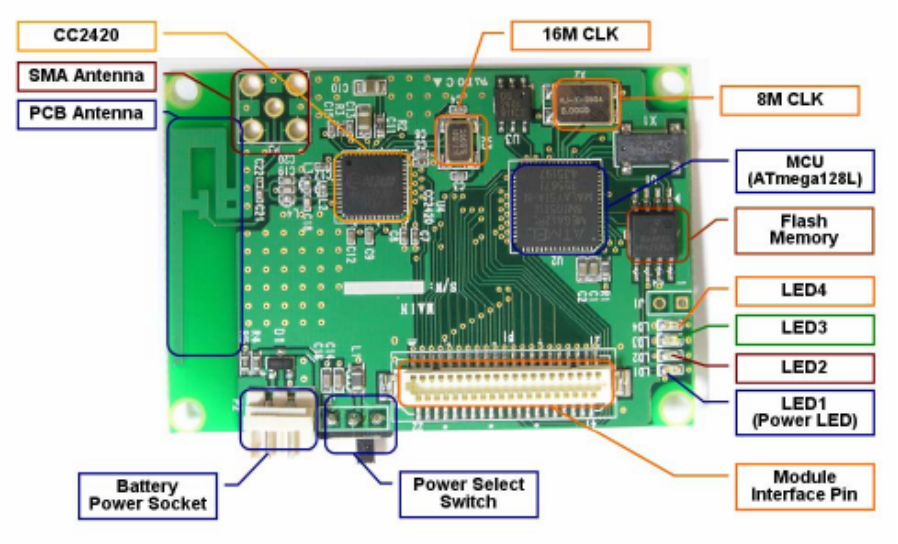
유비쿼터스 센서 네트워크를 위한 경량 프레임워크의 구현은 센서 노드 부분과 서버 부분으로 나누어진다. 먼저 센서 노드의 하드웨어 사양은 <표 5-1>과 같다.

<표 5-1> 센서 노드의 하드웨어 사양

| 항목 | 사양 |
|---|---|
| CPU | ATmega128L , 8bit |
| 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 |




[그림 5-4] 센서 모듈




[그림 5-5] 센서 메인 모듈

그리고 수신용 메인 모듈에 연결하여 데이터를 분석 처리하는 서버의 사양은 <표 5-2>와 같다.

<표 5-2> uSNSM이 동작되는 서버의 사양

| 항목 | 사양 |
|---|---|
| CPU | Pentinm 4, 782MHz, 32bit |
| Memory | 512MB |
| Harddisk Drive | 70 GB |
| Operation System | Windows XP |

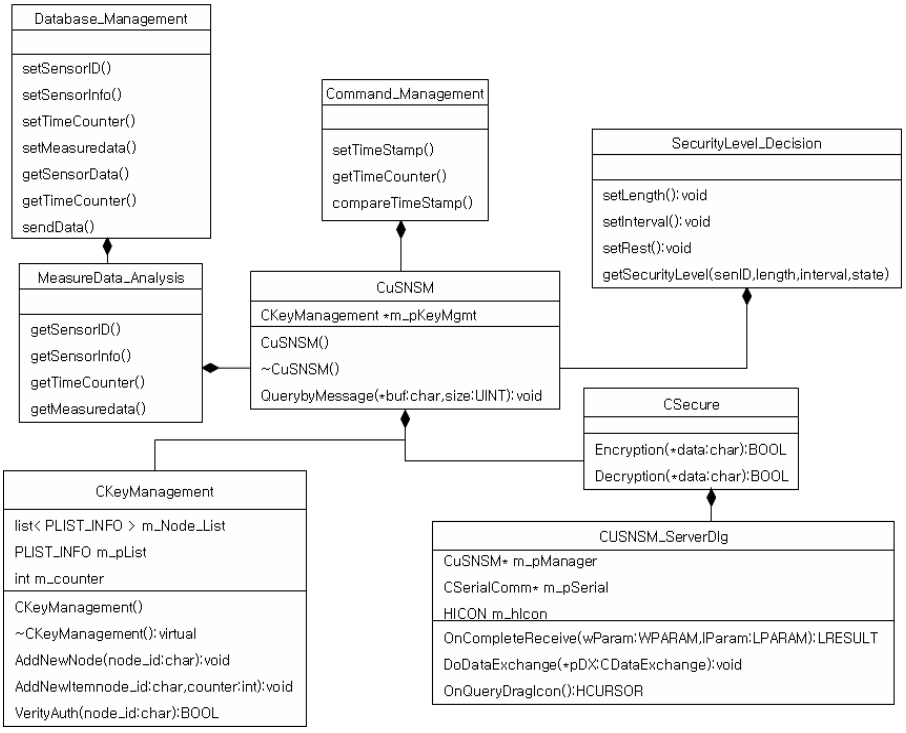
### 5.2.1 서버의 보안 관리자 구현

서버에 포함이 되는 유비쿼터스 센서 네트워크 보안 관리자(uSNSM)는 [그림 5-6]과 같은 모듈 구조를 갖는다. 그리고 서버에 동작이 되는 uSNSM의 구현 환경은 다음 <표 5-3>과 같다.

<표 5-3> uSNSM 구현 환경

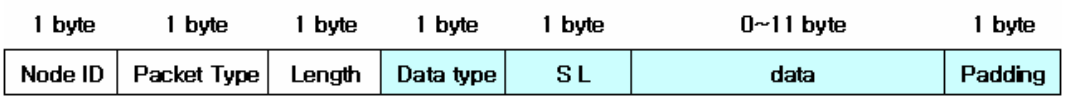
| 항목 | 내용 |
|---|---|
| 실행 환경 | Windows XP |
| 프로그래밍 언어 | C++, MFC |
| Serial Interface | MAX3232 (9600bps) |

[그림 5-6]의 CUSNSM_ServerDlg 클래스에서 먼저 로우 데이터를 처리를 하게 된다. 센서 노드로부터 전달된 데이터를 처리하기 위해서 암호화가 되어 있을 경우 CSecure 클래스를 호출 하여 데이터를 복호화를 한다. 복호화가 끝난 데이터는 CuSNSM 클래스에서 데이터를 분석하여 센서 노드로부터 전달된 데이터 패킷에 대하여 어떠한 행동을 해야 할지 결정을 한다. 센서 노드로부터 전달된 데이터가 단지 센서 노드의 계측 데이터 일 경우 MeasureData_ Analysis 클래스를 호출하여 데이터를 데이터베이스에 저장 할 수 있도록 하며 네트워크의 상태를 감시하기 위한 시큐어 카운터 값은 따로 관리를 한다. 센서 노드에서 전달된 데이터가 인증, 키 교환 등의 보안과 관련된 데이터 일경우 Command_Management 클래스에서 보안을 위한 각각의 함수를 호출 하게 되며 네트워크 상태에 이상이 생기면 SecurityLevel_Decision 클래스에서 센서 및 네트워크 상태에 대한 보안 레벨을 결정하게 된다.




[그림 5-6] uSNSM UML 다이어그램

Qplus에서 지원하는 패킷은 라우팅 데이터 부분을 제외하였을 경우 Node ID, Packet Type, Length, 그리고 Length의 수에 따라 SensorID, ADCH(센서의 ADC 값), ADCL(센서의 ADC값)의 데이터 값이 포함 되고 마지막으로 패딩 값이 포함이 된다. 따라서 센서에서 전송되는 패킷의 값은 최소 7 byte에서 최대 22 byte까지 규격화 되어있다. 하지만 서버에서 센서로 보내는 데이터는 정의가 없고 사용자가 정의해서 사용하도록 되어있다. 본 논문의 유비쿼터스 센서 네트워크를 위한 경량 보안 프레임워크에서는 서버(uSNSM)에서 센서 노드로 가는 패킷의 구조를 [그림 5-7]과 같이 정의 하였다. 서버에서 센서로 보내는 데이터 패킷은 센서와 인증 및 키 교환 할 때 그리고 센서 노드의 보안 레벨을 변경하여 센서 노드의 상태를 변경할 때 전송하게 된다.




Node ID : Sensor 노드별 ID
Packet Type : 패킷의 타입
Length : 데이터에 속한 센서의 수
Data type : 데이터 타입
SL : 보안 관련 명령어
data : 데이터
Padding : 패딩

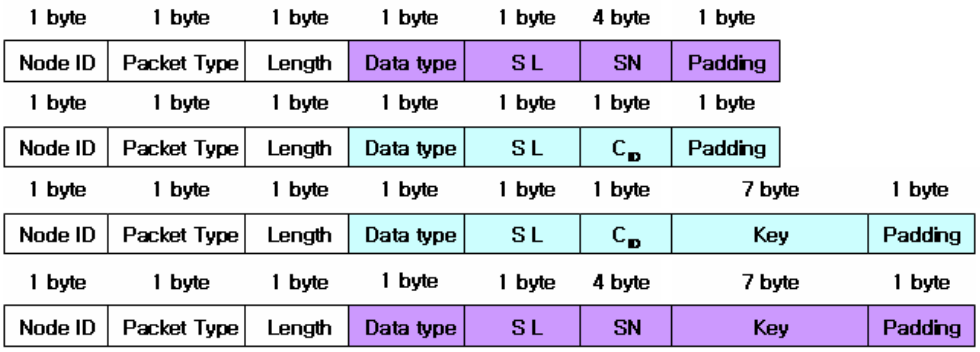
[그림 5-7] uSNSM에서 센서 노드로 보내는 패킷의 구조

[그림 5-7]에서 NodeID, Packet Type, Length 항목은 Qplus에서 사용하는 방식이고 실제 보안 정책을 포함해서 변경되는 부분은 Data type, SL, data, Padding 부분이다. 서버의 uSNSM에서 센서 노드로 보내는 데이터의 구조는 [그림 5-8]과 같이 네 가지 종류가 있다.

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 전달이 되는 부분이다. 공개키 전송의 경우 암호화 하지 않으며 대칭키 전송의 경우 암호화를 하여 전송한다. [그림 7-5]에서 첫 번째와 네 번째 패킷의 경우 정책에 따라서 암호화를 하여 전송하며 두 번째와 세 번째 패킷은 암호화를 하지 않는다. 암호화를 하지 않는 패킷의 크기는 7 byte, 14 byte의 두 가지 종류가 있고 패킷의 전체 크기는 7 byte, 10 byte, 14 byte, 17 byte 가 된다. 암호화를 하는 패킷의 크기는 7 byte, 14 byte 두 가지이며 이 패킷의전체크기는 10 byte, 17 byte가 된다.

센서의 ID는 센서 네트워크 설계자가 센서 네트워크를 설계를 하며 해당 센서의 위치나 센싱 데이터에 따라서 임의의 정수 값으로 결정하게 된다. 실제 보안 모듈이 센서 노드에 포함이 될 때 프로그래머에 의해서 정해진 값이 포함이 된다.



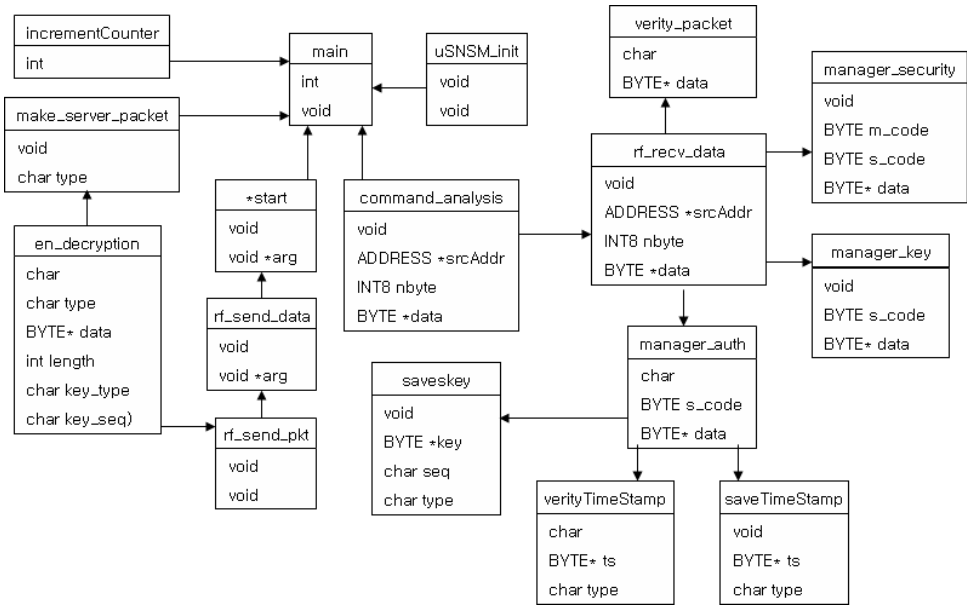

Data type : 데이터 타입
SL : 보안 관련 명령어
data : 데이터
Padding : 패딩
SN : 시큐어 넘버 정보
Key : 암호키
$C_{ID}$ : 센서 ID

[그림 5-8] uSNSM에서 전송하는 패킷의 종류

### 5.2.2 센서 노드 경량 보안 관리자 구현

센서 노드에 포함이 되는 경량 보안 관리자(LWSM)는 [그림 5-9]와 같은 모듈 구조를 갖는다. 센서 노드에 동작이 되는 경량 보안 관리자의 구현 사양은 다음 <표 5-4>와 같다.

[그림 5-9]의 그림은 센서 노드에 포함이 되는 경량 보안 관리자 모듈의 구조를 나타낸 것이다. 경량 보안 관리자의 경우 전송하려는 데이터 처리와 전송 받은 데이터 처리 두 가지로 구분 할 수 있다. 전송하려는 데이터는 주로 암호화를 하여 데이터를 전송하는 모듈로 이루어져 있다. 서버로부터 전송 받은 데이터의 경우 인증, 키 교체에 관한 프로토콜 관련 명령어들과 보안 레벨을 변경하려는 명령어들로 구분된다. 서버로부터 데이터가 수신이 되면 command_analysis 모듈에서 데이터 분석을 하여 데이터 처리를 시작한다. 서버에서 전송된 해당 명령어에 따라서 인증, 키 교체, 그리고 센서 노드의 동작 제어를 처리 하게 된다.




[그림 5-9] 경량 보안 관리자 모듈 구조

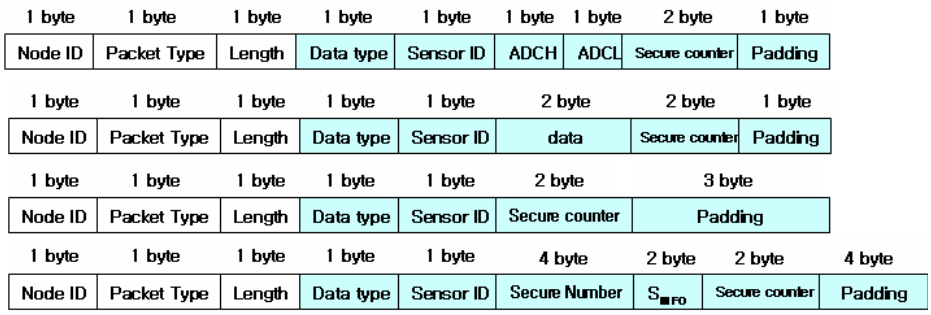
센서 노드에서 전송 하는 패킷의 구조는 [그림 5-10]과 같다. [그림 5-10]에서 Node ID, Packet Type, Length 부분은 Qplus에서 지정한 패킷의 구조이다. 그리고 Data type, Secure Counter, $C_{INFO}$, Secure Number 등의 부분은 새로 정의된 보안 관련 구조이다. 실제 Node ID, Packet Type, Length 부분을 제외한 나머지 부분만 암호화 처리가 된다. 센서의 정보 부분은 실제 센서의 CPU에서 센서 노드 identification 레지스터에서 값을 가져와서 사용하게 된다. 센서 노드의 identification 레지스터에 있는 값 중에서 Part Number의 일부와 제조 ID의 값의 합으로 2byte를 가져오기 때문에 제조 ID만 알고 있더라도 유추해내기가 어렵다. 센서 ID의 경우는 센서 네트워크 설계자에 의해서 결정된 정수 값으로 프로그래머가 직접 입력해야 하는 값으로 센서 정보와는 차이가 있다.

본 논문에서 제안하는 유비쿼터스 센서 네트워크를 위한 경량 보안 프레임워크의 센서 노드에서 보안 정책이 포함된 패킷은 네 가지로 구분이 된다. 그 중 센서 노드에서 주변 환경 데이터를 수집해서 보내는 기본 패킷은 7 byte 이며, 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.

<표 5-4> LWSM 구현 환경

| 항목 | 내용 |
|---|---|
| 실행 환경 | Qplus |
| 프로그래밍 언어 | C |
| RF | CC2420 |



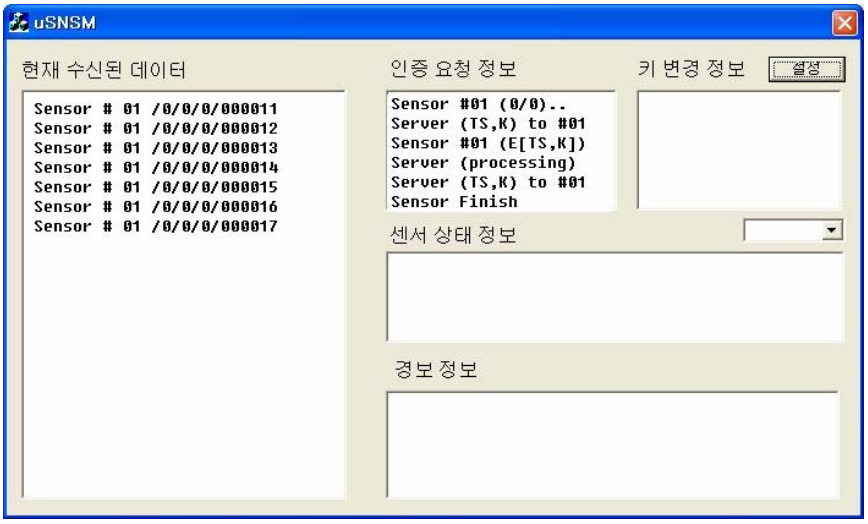

Node ID : Sensor 노드별 ID
Packet Type : 패킷의 타입
Length : 데이터에 속한 센서의 수
Data type : 데이터 타입
Sensor ID : 센서 지정 ID
Secure Number: 시큐어 넘버
$S_{\blacksquare FO}$ : 센서 정보
Secure counter: 센서 시큐어 카운터
data : 요구 데이터
Padding : 패딩

[그림 5-10] 센서 노드에서 전송되는 패킷의 구조

### 5.2.3 유비쿼터스 센서 네트워크 감시 구현

유비쿼터스 센서 네트워크 감시는 인증, 암호화의 수동적인 보안 정책뿐만 아니라 현재 네트워크의 상태를 파악해서 능동적으로 공격자의 공격에 대처 할 수 있게 도

움을 주기 때문에 보안 프레임워크를 구성하는데 필수적이다. 본 논문에서 지원하는 유비쿼터스 센서 네트워크를 위한 경량 보안 프레임워크에서는 센서 노드로부터 서버로부터 전송되는 데이터는 유비쿼터스 센서 네트워크 보안 관리자(uSNSM)가 데이터 분석하여 모니터링하게 된다. 센서로부터 전송되는 데이터는 크게 일반 데이터와 보안 관련 데이터로 구분하게 된다. 센서 노드로부터 전송되는 모든 데이터는 암호화가 되어 있기 때문에 복호화를 한 후 센서 ID 값과 Data type 값이 허용범위를 벗어나게 되면 모니터링 모듈에서 처리를 하게 된다. 또한 인증된 센서 노드로부터 수신된 일반 데이터는 시큐어 카운터의 기록이 자동적으로 기록이 되어 시큐어 카운터의 값의 변동 폭이 커질 경우 모니터링 모듈에서 처리하게 된다. 또한 인증, 키 교환 데이터의 경우에도 총 전송 시간, 시큐어 카운터 값을 모니터링하여 확인을 하도록 하여 현재 센서 노드와 서버 간의 네트워크 상태 및 공격자의 공격 여부를 확인 할 수 있다.




[그림 5-11] 센서 노드 감시 프로그램

[그림 5-12]는 서버 감시 프로토콜 중에서 일반 데이터 감시 프로토콜이다. 평상시에 센서 노드로부터 오는 모든 데이터를 감시 하여 보안 레벨 상태를 결정하게 된다.

```
1  if (preSecureCounter =! SecureCounter+1)
2                  if (exist SecureCounter) send the risk message
3                  if (SecureCounter – preSecureCounter = 2)  send the note message
4                  else send the warning message
5  else if (data interval != permissible range) send the warning message
6  else if (data != permissible range) send the warning message
```

[그림 5-12] 일반 데이터 수신 시 감시 프로토콜

센서 노드에서 생성하여 패킷에 포함하는 시큐어 카운터의 연속성 여부 혹은 데이터의 손실 여부, 지연 여부 등을 확인하여 4.3.1절의 주의, 경고, 위험의 세 가지 보안 단계를 결정하게 된다.

[그림 5-13]은 인증 및 키 교체 관리 프로토콜 실행 시 수신 되는 모든 데이터를 감시하는 프로토콜이다.

```
1  if (preSecureCounter =! SecureCounter+1)
2                  if (exist SecureCounter) send the risk message
3                  if (SecureCounter – preSecureCounter = 2)  send the risk message
4  if (completion time != permissible range) send the risk message
5  if (SecureNumber != storeSecureNumber) send the risk message
```

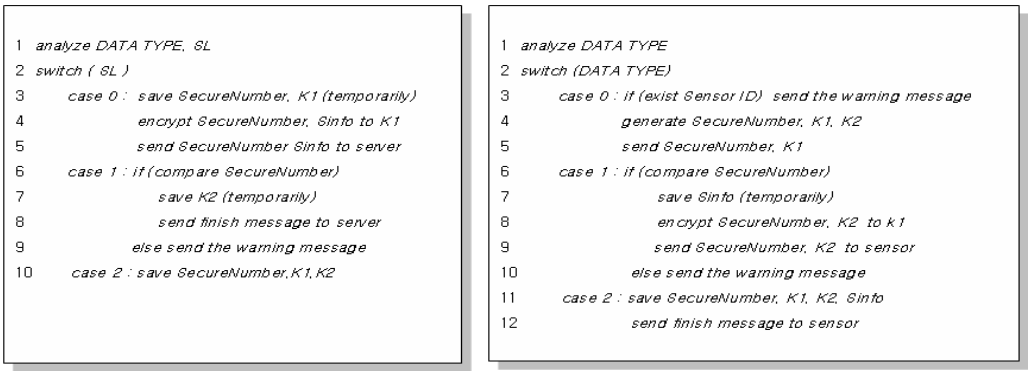
[그림 5-13] 인증, 키 교체 및 관리 진행시 감시 프로토콜

센서 노드에서 전송되는 패킷에 포함되어 있는 시큐어 카운터 값뿐만 아니라 시큐어 넘버 및 프로토콜이 종료되는 시간까지 감시하여 인증의 허용 여부, 키 교체의 허용 여부 및 공격자의 공격 상태 등을 파악 할 수 있다.

### 5.2.4 더블키를 이용한 인증 프로토콜 및 키 교환 알고리즘 구현

유비쿼터스 센서 네트워크를 위한 경량 보안 프로토콜에서 사용하는 두 가지 인증 프로토콜 중에서 대칭키 기반의 더블키를 이용한 인증 프로토콜은 [그림 5-14]와 같다. [그림 5-14]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 5-14]에서 왼쪽의 센서 노드에서 동작하는 프로토콜의 첫줄에서 서버로부터 온 데이터에서 Data type과 SL 부분을 분석한다. 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.



```
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
```

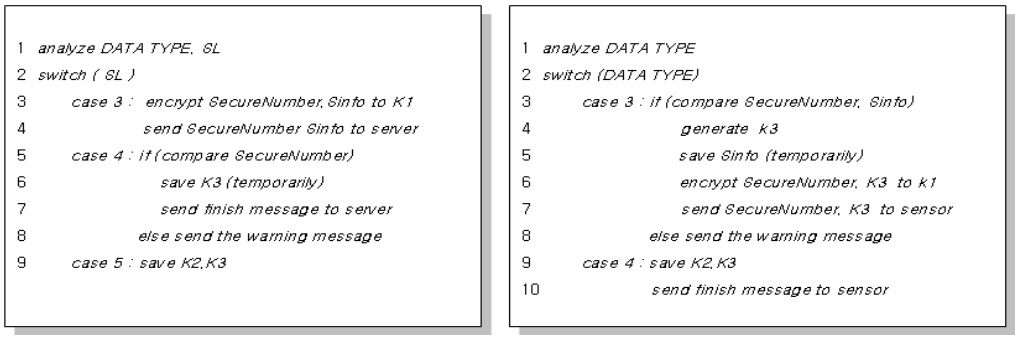
```
1  analyze DATA TYPE
2  switch (DATA TYPE)
3       case 0 : if (exist Sensor ID)  send the warning message
4                generate SecureNumber, K1, K2
5                send SecureNumber, K1
6       case 1 : if (compare SecureNumber)
7                    save SInfo (temporarily)
8                    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 5-14] 서버와 센서 노드의 더블키 인증 프로토콜

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 대칭키를 받고 첫 번째 대칭키를 이용하여 데이터를 암호화 하여 서버로 전송하

게 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 지정 정보까지 저장하게 된다.

대칭키 기반의 보안 정책의 경우 키 분배가 가장 중요한 사항이다. 유비쿼터스 센서 네트워크의 경우 센서에서 사용하는 대칭키를 주기적으로 바꿔주는 것이 센서 네트워크를 안전하게 유지시키는데 도움이 된다. 유비쿼터스 센서 네트워크를 위한 경량 보안 프로토콜에서 대칭키 기반의 보안 정책 중에서 키 교체 프로토콜은 [그림 5-15]와 같다.




[그림 5-15] 서버와 센서 노드의 더블키 교체 프로토콜

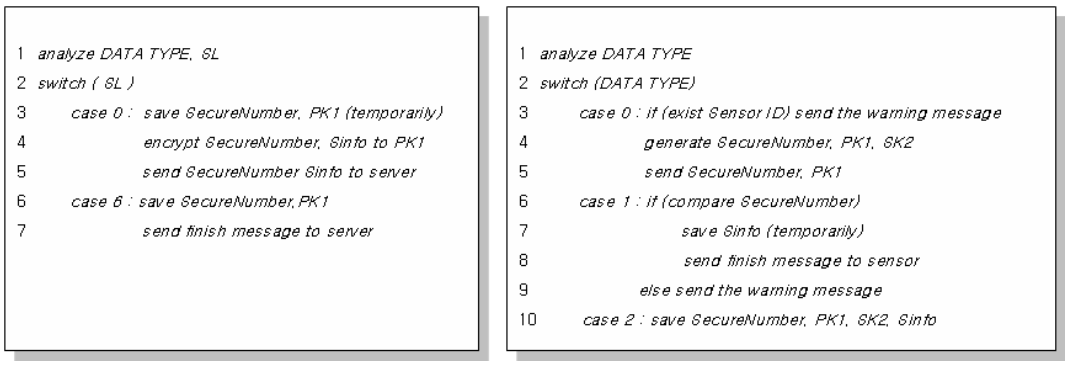
[그림 5-15]에서 왼쪽의 프로토콜은 센서 노드에 포함되어 있는 대칭키 교체 프로토콜이며 오른쪽의 프로토콜은 서버에 대칭키 교체 프로토콜이다. 서버로부터 대칭키 교체 신호가 오면 센서 노드 교체 프로토콜의 세 번째 줄에서 기존의 사용하는 대칭키로 데이터를 암호화 하여 전송하게 된다. 전송된 데이터는 서버 교체 프로토콜의 세 번째 줄에서 데이터를 비교하게 된다. 교체 신호를 보낸 적이 없는데 데이터가 왔거나 데이터가 다를 경우 보안 관리자 모듈에 경고 메시지를 전송하게

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.

### 5.2.5 공개키를 이용한 인증 프로토콜 및 키 교환 알고리즘 구현

유비쿼터스 센서 네트워크를 위한 경량 보안 프로토콜에서 사용하는 두 가지 인증 프로토콜 중에서 공개키를 이용한 인증 프로토콜은 [그림 5-16]과 같다.

[그림 5-16]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.

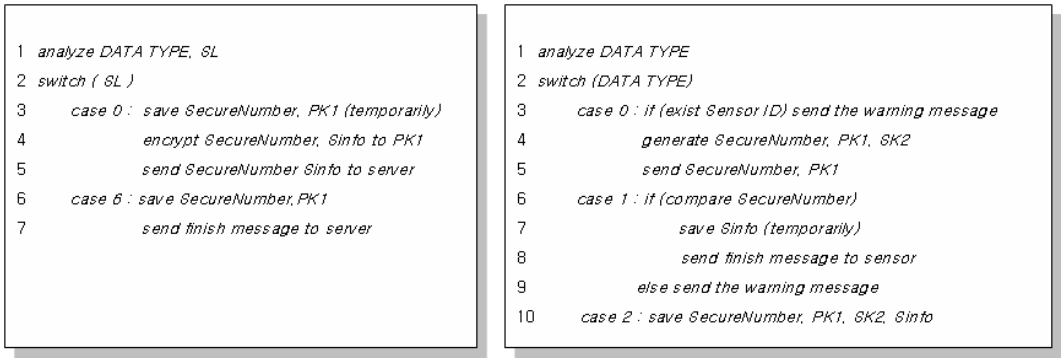



[그림 5-16] 서버와 센서 노드간의 공개키를 이용한 인증 프로토콜

[그림 5-16]에서 센서 노드로부터 인증 요청이 들어오면 서버 인증 프로토콜 세

번째 줄에서 센서의 정보를 확인한다. 만일 이미 인증이 된 센서 노드거나 센서 노드의 값이 허용치 이상 일 경우 보안 관리자 모듈로 경고 메시지를 보내게 된다. 네 번째, 다섯 번째 줄에서는 시큐어 넘버와 비밀키, 공개키를 생성하여 시큐어 넘버와 공개키를 센서 노드로 보내게 된다. 센서 노드 인증 프로토콜에서 세 번째 줄에서 서버로부터 온 데이터에서 공개키를 이용하여 시큐어 넘버와 센서 정보를 암호화하여 서버로 보내게 된다. 서버 인증 프로토콜의 여섯 번째 줄에서 데이터를 비교하여 센서 노드로부터 온 데이터를 저장하거나 만일 값이 다를 경우 아홉 번째 줄에서 보안 관리자 모듈로 경고 메시지를 보내게 된다. 서버 인증 프로토콜의 열 번째 줄과 센서 노드의 여섯 번째 줄에서처럼 프로토콜의 종료 신호를 서로 전송하게 되면 각각의 데이터를 저장하게 된다.

공개키 기반의 보안 정책의 경우 키 분배는 쉽게 이루어 질수 있지만 유비쿼터스 센서 네트워크의 환경에서 서버와 센서 노드간의 종속적인 관계에서 공개키 기반의 보안 정책을 이용할 경우 키 분배는 대칭키 분배만큼 중요한 사항이 된다. 아무리 공개키를 사용하더라도 공개키를 주기적으로 교체해 주는 것이 센서 네트워크를 안전하게 유지시킬 수 있다. 유비쿼터스 센서 네트워크를 위한 경량 보안 프로토콜에서 공개키 기반의 보안 정책 중에서 키 교체 프로토콜은 [그림 5-17]과 같다.




[그림 5-17] 서버와 센서 노드간의 공개키 교체 프로토콜

[그림 5-17]의 왼쪽 프로토콜은 서버에서 센서 노드로 전송된 데이터를 처리하는

교체 프로토콜이며 오른쪽은 센서 노드로부터 서버로 전송된 데이터를 처리하는 교체 프로토콜이다. 공개키 교체 프로토콜은 서버의 교체 신호로부터 시작된다. 센서 노드 교체 프로토콜의 세 번째 줄에서 센서에서 서버로부터 공개키 교체 신호를 받게 되면 서버로 기존의 사용하던 공개키를 이용하여 데이터를 암호화 하여 전송하게 된다. 서버의 교체 프로토콜 세 번째 줄에서 해당 데이터를 분석하여 새로운 공개키와 비밀키를 생성하여 센서로 공개키를 전송하게 된다. 만일 요청하지 않은 데이터가 전달되면 보안 관리자 모듈로 경고 메시지를 전송하게 된다. 센서 노드의 교체 프로토콜에서 다섯 번째 줄에서 서버로부터 전달 받은 공개키를 이용하여 다시 데이터를 암호화 하여 서버로 전송하게 된다. 서버의 교체 프로토콜 일곱 번째 줄에서 데이터를 확인하고 만일 값이 맞지 않거나 요청하지 않은 데이터가 왔을 경우 보안 관리자 모듈로 경고 메시지를 전송하게 된다. 서버와 센서 노드간의 교체 프로토콜이 각각 종료 메시지를 보내게 되면 저장 데이터를 각 노드에서 저장을 하고 프로토콜이 종료가 된다.

# 제 6 장 성능 평가

기존의 센서 네트워크에서 보안 정책은 대칭키 기반의 보안 정책이 대부분을 차지하고 있다. 센서 네트워크의 대칭키 기반 보안 정책은 데이터의 기밀성, 무결성에 중점을 두고 있기 때문에 대칭키 기반 보안 정책의 성능 평가는 보안 정책이 포함되었을 경우와 보안 정책이 없을 경우를 비교하여 센서 노드의 처리량 증가 및 소비되는 에너지 그리고 MCU의 처리 시간과 같은 항목에 대한 성능 평가를 하고 있다. 센서 네트워크에서 공개키 기반의 보안 정책의 경우는 대칭키 보안 정책에 비하여 상대적으로 많은 처리량과 에너지 소비 때문에 센서 네트워크에 적용하지 못하고 있는 실정이었다. 하지만 최근 들어 타원 곡선 암호 알고리즘을 중심으로 센서 네트워크 보안 정책에 공개키 방식 적용 연구가 활발히 진행이 되고 있다. 공개키 기반의 암호 정책의 경우는 대칭키 보안 정책에 비하여 상대적으로 많은 처리량과 에너지 소비량 때문에 성능 평가 항목으로 저전력, 경량화 등에 중점을 두고 센서 네트워크 적용에 대한 가능성에 대한 성능 평가를 하고 있다.

본 논문은 유비쿼터스 센서 네트워크의 센서 노드의 제약적인 환경에 맞는 인증 프로토콜과 키 교체 프로토콜 그리고 센서 네트워크 감시 프로토콜을 대칭키와 공개키 기반의 보안 정책으로 나누어서 제안하였다. 본 논문에서 제안하는 대칭키 기반의 보안 정책은 기존의 대칭키 보안 정책에는 포함되지 않는 인증 과 키 교체 프로토콜을 포함하였으며 공개키 기반의 보안 정책은 기존의 공개키 보안 정책의 인증, 키 생성, 관리 모듈을 제외하였으나 새로운 인증, 키 관리 프로토콜을 포함하였으며 기존의 보안 정책에 비하여 저전력, 경량화에 중점을 두고 설계를 하였다. 경량화 된 암호화 및 복호화 알고리즘은 기존의 방식을 이용하였기 때문에 암호화 처리 시간, 암호화 알고리즘 사용 시 추가 전력, 암호화 연산량에 대한 항목은 제외를 하며 본 논문에서 제안하는 보안 정책이 포함된 후의 센서 노드의 저전력과 센서 노드에 포함되는 프로토콜의 경량화에 중점을 두고 성능 평가를 논의 한다.

본 논문에서 제안하는 유비쿼터스 센서 네트워크를 위한 경량 보안 프로토콜은 센서 노드의 저전력과 보안 프로토콜의 경량화에 중점을 두고 설계를 하였기 때문

에 제안하는 보안 프로토콜의 타당성을 검증하기 위해서 다음과 같은 성능평가 항목을 설정 하였다. 첫째, 센서 노드의 소비 전력은 약 40%가 센서 노드의 동작에 소비가 된다. 따라서 보안 정책이 포함된 후의 전송 되는 패킷 크기의 증가율은 센서 노드의 동작과 관련이 있기 때문에 성능 평가에 중요한 항목이 될 것이다. 둘째, 센서 노드의 소비 전력의 약 60%가 패킷을 전송하고 수신할 때 소비가 되기 때문에 새로운 프로토콜의 포함되는 본 논문의 보안 프로토콜에서는 중요한 성능 평가 항목으로 측정해야 할 것이다. 셋째, 센서 노드의 처리량은 센서 노드에 포함되는 보안 모듈의 크기에 영향을 미치게 되며 이러한 점은 전력 소비와도 연관 관계가 있다. 따라서 센서 노드에 포함되는 일반적인 보안 정책의 크기와 본 논문에서 제안하는 센서 노드에 포함되는 보안 프로토콜의 크기를 비교하여 어느 정도의 경량화가 이루어졌는지에 대한 항목도 중요한 성능 평가 요소가 될 것 이다.

본 논문에서 제안한 유비쿼터스 센서 네트워크를 위한 경량 보안 프로토콜의 타당성을 검증하기 위해서 다음과 같이 세 가지 항목에 대하여 성능 평가 항목을 설정하였다.

- 암호화 모듈 적용 후 패킷 크기 증가율
- 인증, 키 교체 알고리즘 실행 시 추가적인 패킷 전송량 증가
- 센서 노드에 포함되는 대칭키 및 공개키 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 전송 되는 패킷의 양이 늘어나기 때문에 전력 소비 증가는 피할 수 없을 것이다. 따라서 성능 평가에서

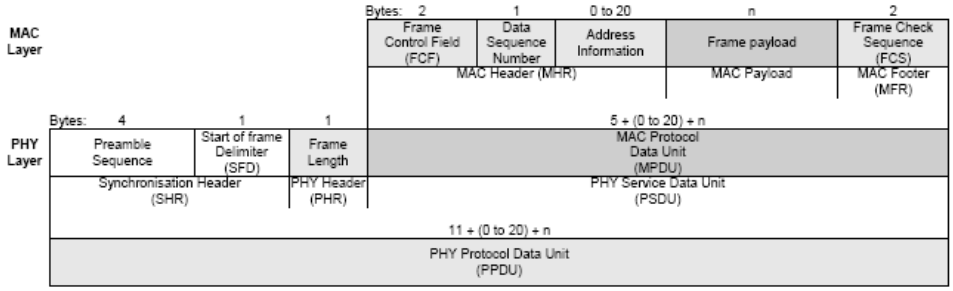
중점적으로 볼 것은 일반적인 상태에서와 기존의 보안 정책이 포함되었을 경우, 그리고 본 논문에서 제안하는 경량 보안 프레임워크의 공개키 및 대칭키 기반의 보안 정책이 포함되었을 경우의 상태를 비교하여 소비되는 전력의 상대적인 차이이다. 또한 기존의 암호화 및 복호화 정책이 모두 포함된 경우와 제안한 프로토콜을 삽입하고 불필요한 모듈을 제외 했을 경우의 차이로 성능 평가를 비교하여 본다. 암호화 모듈 적용 후 패킷 크기 증가율에 대한 성능 평가는 6.1절에서 논의하고, 인증, 키 교체 알고리즘 실행 시 추가적인 패킷 전송량 증가에 대한 성능 평가는 6.2절에서 논의 한다. 마지막으로 센서 노드에 포함되는 대칭키 및 공개키 기반의 보안 모듈과 일반 보안 정책의 모듈 크기 비교 항목은 6.3절에서 평가하도록 한다.

## 6.1 암호화 모듈 적용 후 패킷 크기 증가율

본 절에서 성능 평가로 비교하여 볼 것은 암호화 정책이 포함된 후 패킷의 크기 변화 이다. 본 논문의 구현 환경인 RF CC2420에서 사용하는 패킷의 구조는 [그림 6-1]과 같다[16]. [그림 6-1]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 6-1]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 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 정책과 공개키 기반의 보안 정책의 패킷의 구조는 [그림 5-8]과 [그림 5-10]에서 설명하였다. 대칭키 기반의 더블키를 이용한 보안 정책의 경우 센서 노드에서 전송 되는 패킷 데이터 크기는 보안 관련 데이터 3 byte를 포함하여 최소 10 byte이며 인증, 키 교체 시 패킷의 크기는 14 byte를 사용한다. 센서 노드의 패킷 수신 데이터의 크기는 8byte

와 24 byte로 주로 인증, 키 교체 및 보안 레벨 조정 등에서 사용 한다. 공개키 기반의 보안 정책의 경우 센서 노드에서 전송 되는 패킷 데이터 크기는 보안 관련 데이터 3 byte를 포함하여 10 byte 이며 인증 , 키 교체 시 사용되는 패킷의 크기는 14 byte를 사용한다. 공개키 기반의 보안 정책의 센서 노드의 수신 되는 패킷의 크기는 8 byte와 17 byte로 주로 인증과 키 교체 시 사용된다.




[그림 6-1] IEEE 802.15.4 프레임 구조

그 중 인증, 키 교체의 패킷을 제외한 센서 노드에서 센싱한 데이터를 기본 패킷을 기준으로 보안 정책이 포함된 후 패킷의 증가량을 보면 <표 6-1>과 같다.

<표 6-1> 센서 기본 패킷에 보안 정책 포함 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 | |
| 대칭키 정책 포함 시 | 11+(0+20) | 3 | 10 | 21 | 41 | 16.7, 7.9 |
| 공개키 정책 포함 시 | 11+(0+20) | 3 | 10 | 21 | 41 | 16.7, 7.9 |

<표 6-1>에서 보면 최소 크기의 데이터 패킷일 경우 16.7%의 증가율을 가져왔으며 패킷의 크기가 최대 일 경우 7.9%의 증가율을 보였다. TinyOS를 기반으로 연구중인 TinySec, SNEP, MiniSec의 경우 증가율을 비교하면 다음 <표 6-2>와 같다

[17].

본 논문의 시뮬레이션은 RF CC2420 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 패킷량의 증가는 실제로 센서 노드에서 데이터를 전송 할 때 128 byte 까지는 전송 전력은 소비되지 않는다. 따라서 <표 6-2>의 증가율은 센서 노드에서 패킷 제작을 위한 CPU 연산에만 전력이 추가 적으로 소비가 될 뿐 전송 시에는 같다고 볼 수 있다.

<표 6-2> 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 | - |
| 제안한 보안 프로토콜 | 38 | 3 | 41 | 7.9 |

<표 6-2>에서 비교는 최대 패킷의 크기로만 이용하여 비교 하였다. <표 6-2>에서 알 수 있듯이 본 논문에서 제안한 유비쿼터스 센서 네트워크를 위한 경량 보안 프로토콜에서 사용하는 보안 프로토콜은 다른 보안 정책들에 비하여 가장 낮은 패킷 증가율을 보이고 있다. 본 논문에서 구현을 Qplus에서 했지만 제안하는 보안 프로토콜은 센서 노드의 운영체제와 무관하게 실행 할 수 있기 때문에, TinyOS 위에서 동작을 할 경우 패킷의 증가율은 <그림 6-2>의 MiniSec와 같은 8.3 %가 될 것이다. 그럴 경우에도 기존의 보안 정책에 비교하여 가장 낮은 증가율을 보이고 있다고 할 수 있다.

## 6.2 제안한 프로토콜 실행 시 추가적인 패킷 전송량 증가

본 절에서 비교 할 항목은 인증, 키 교체 시 추가 패킷과 전송량 증가에 관한 부분이다. 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 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.

<표 6-3> 인증, 키 교체 보안 정책 후 패킷 증가 비교 (단위 byte)

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

본 논문에서 제안하는 유비쿼터스 센서 네트워크를 위한 경량 보안 프레임워크에서는 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 포함 되었을 경우와 보안 패킷 전송 방식과 직접적인 비교는 불가능 하지만 본 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 6-3>과 같다.

송수신 칩의 전송과 수신시 소비되는 전력이 차이가 나서 <표 6-3>과 같이 전송과 수신으로 나누어서 계산을 하였다. 대칭키 보안 정책의 경우 키의 크기가 56 bit, 112 bit 두 가지가 가능하며 공개키의 경우 56 bit 하나만 지원하기 때문에 수신 될 때의 패킷의 크기가 공개키 보안 정책에 비해서 약간 더 크다. 본 논문에서 제안하는 보안 정책에서 인증과 키 교환 시 패킷의 크기는 128 byte를 넘지 않기 때문에 한 번 전송되는 패킷에 소비되는 전력은 일정하다.

제안한 프로토콜을 사용할 시 기존의 보안 프로토콜에 없는 추가 전력을 사용하게 된다. 제안한 프로토콜은 인증과 키 교체 관리 프로토콜로 이루어져 있기 때문에 인증 프로토콜 실행 시 소비되는 전력을 A, 키 교체 프로토콜 실행 시 소비되는 전력을 K라고 한다면 추가 소비 전력을 P는 다음과 같이 계산 할 수 있다.

$$P = A + K \qquad \text{(식 6-1)}$$

사용 횟수를 t, 전송 횟수를 st, 전송 소비 전력을 sc, 수신 횟수를 rt, 수신 소비 전력을 rc 라고 한다면 인증 프로토콜 사용 시 소비 되는 전력은 (식 6-2)과 같이 표현 할 수 있다.

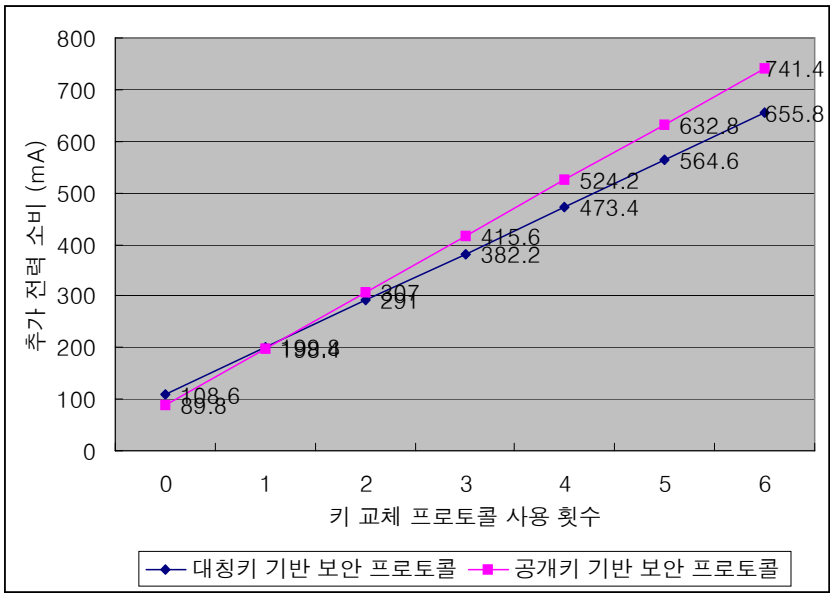
$$A = at \times ( ast \times sc + art \times rc ) \qquad \text{(식 6-2)}$$

또한 키 교체 프로토콜 사용 시 소비 되는 전력은 (식 6-3)과 같이 된다.

$$K = kt \times ( kst \times sc + krt \times rc ) \qquad \text{(식 6-3)}$$

본 논문의 구현 하드웨어 사양에서 송수신을 담당하는 칩의 경우 전송 시 17.4mA 그리고 수신 시 18.8mA를 소비하게 된다. 따라서 제안하는 대칭키 기반 보

안 프로토콜에서 인증 프로토콜은 한번만 하도록 되어있기 때문에 (식 6-2)에서 A의 값은 108.6mA가 된다. 또한 대칭키 기반의 키 교체 프로토콜을 한번 실행 시(kt = 1) 소비되는 전력 K는 91.2 mA가 된다. 마찬가지로 공개키 기반의 보안 프로토콜 사용 시 소비되는 추가 전력은 인증 프로토콜 사용 시 89.8 mA, 키 교체 프로토콜 사용 시 108.6 mA가 된다. (식 6-2)와 (식 6-3)에 의해서 대칭키 기반의 보안 프로토콜 사용 시 소비 되는 추가 전력과 공개키 기반의 보안 프로토콜 사용 시 소비되는 추가 전력은 [그림 6-2]와 같이 나타낼 수 있다.




[그림 6-2] 보안 프로토콜 사용 시 소비되는 추가 전력

[그림 6-2]에서 보면 대칭키 기반의 보안 프로토콜은 공개키 기반의 보안 프로토콜보다 인증 프로토콜 사용 시 소비되는 전력은 더 많다. 하지만 키 교체 프로토콜에 사용되는 추가 소비 전력은 공개키 기반의 보안 프로토콜에서 더 많이 소비하기 때문에 키 교체 프로토콜이 네 번 이상 실행이 된다면 공개키 기반의 보안 프로토콜에서 소비되는 추가 소비 전력은 대칭키 기반의 보안 프로토콜에 비해서 커지게 된다. 하지만 실제 센서 노드에 제한적인 전력량을 감안하더라도 인증 프로토콜과

다섯 번의 키 교체 프로토콜을 실행 할 경우 소비되는 전력은 대칭키 기반 보안 프로토콜의 경우 473.4mA, 공개키 기반의 보안 프로토콜의 경우 524.2mA가 추가적으로 소비하도록 되어서 제안된 전력량에 크게 영향을 미치는 소비 전력은 아니라고 볼 수 있다.

## 6.3 센서 노드에 포함되는 모듈 크기 비교

본 절에서 비교해 볼 내용은 센서 노드에 포함되는 대칭키 및 공개키 기반의 보안 모듈과 일반 보안 정책의 모듈의 크기를 비교하는 것이다. 만일 기존의 보안 정책을 센서 노드에 포함을 한다면 센서 노드에 포함이 되는 모듈은 대칭키 기반의 정책일 경우 키 생성 모듈, 키 관리 모듈, 암호 및 복호화 모듈, 키 교환 모듈이 반드시 포함이 되어야 하며 공개키 기반의 정책일 경우 키 생성 모듈, 인증 관리 및 처리 모듈, 암호 및 복호화 모듈이 포함이 되어야 할 것이다. 하지만 제안하는 보안 정책의 대칭키 기반의 더블키를 이용한 인증, 키 교환 프로토콜은 암호 및 복호화 모듈, 초기 인증 모듈, 키 교환 모듈만 포함이 되며 공개키를 이용한 인증, 키 교환 프로토콜은 암호화 모듈, 초기 인증 모듈, 키 교체 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. 공개키 기반의 보안 모듈을 사용할 경우 기존의 보안 모듈을 모두 포함 한다면 71828 byte 이상 포함이 되어야 한다. 하지만 본 논문에서 제안하는 공개키 기반의 보안 프로토콜을 사용할 경우 50989 byte 만 포함하면 활용이 가능하였다. 대칭키 기반의 더블키 보안 정책일 경우 약 38.9%의 코드가 제외되었으며 공개키 기반의 보안 정책일 경우 약 29.0%의 코드가 제외되었다. 특히 공개키 기반의 보안 정책일 경우 복호화 작업이 제외되었기 때문에 기존의 공개키 기반의 보안 정책을 사용 할 때에 비하여 센서 노드의 처리량은 2/3 이상 감

소되었을 것으로 측정 되었다.

본 논문에서 제안하는 더블키 보안 프로토콜은 센서의 운영체제, 패킷의 라우팅 방식, 암호 및 복호화 알고리즘에 독립적인 프로토콜이기 때문에 경량화 대칭키 암호 알고리즘을 사용하는 어떠한 정책보다도 38%의 코드 절감 효과를 가져 올수 있으며 센서 노드의 전력 및 처리량을 줄일 수 있다. 그리고 현재는 공개키 기반의 보안 프로토콜 자체의 크기와 암호화 처리 속도의 차이 때문에 대칭키 기반의 보안 프로토콜에 대한 연구가 대부분이지만 유비쿼터스 센서 네트워크의 환경 상 공개키 기반의 보안 정책의 장점이 많기 때문에 3.3절에서 논의했듯이 5년 후에는 공개키 기반의 보안 정책이 주를 이룰 것이다. 공개키 보안 정책을 사용하더라도 본 논문에서 제안한 보안 프로토콜을 사용할 경우에는 약 30 %의 코드 절감 효과와 센서 노드의 소비 전력 및 처리량을 줄일 수 있을 것으로 기대된다.

# 제 7 장 결 론

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 방식에 종속적이지 않은 경량 보안 프레임워크를 제안하였다. 제안한 경량 보안 프레임워크는 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 두고 있다.

본 논문에서 제안한 경량 보안 프레임워크의 특징은 유비쿼터스 센서 네트워크의 제한된 센서 노드에 적합한 보안 알고리즘을 선택하며 센서 노드별로 다른 보안 알고리즘이나 키를 관리 할 수 있도록 지원하였다. 선택이 가능한 보안 프로토콜은 유비쿼터스 센서 네트워크의 센서 노드의 제한된 환경을 고려한 더블키 기반의 경

량 보안 프로토콜과 공개키를 사용한 보안 프로토콜로 구성이 된다. 또한 시큐어 카운터를 이용한 보안과 전송 거리 조절을 이용한 보안 정책을 이용하여 공격자의 공격에 능동적으로 대처를 할 수 있게 지원하며 유비쿼터스 센서 네트워크 상태를 모니터링 하여 현 센서 네트워크 상태에 맞는 보안의 적절한 레벨을 선택하고 관리할 수 있도록 지원 하였다.

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. 또한 모니터링을 통하여 현재의 센서 노드의 상태나 네트워크의 상태를 파악할 수 있도록 하여 안전한 네트워크 상태를 유지할 수 있도록 하였으며 공격자의 공격에 능동적으로 대처할 수 있도록 하여 방어적인 센서 노드를 구성 하였다. 제안한 유비쿼터스 센서 네트워크를 위한 경량 보안 프레임워크는 스마트 오피스 및 스마트 홈과 같은 실제 환경의 유비쿼터스 센서 네트워크에 적합한 보안 프레임워크라고 할 수 있다.

향후 연구 과제로는 센서 노드의 하드웨어 성능이 향상 되는 것을 예상하여 센서 노드에 다른 암호화 복호화 알고리즘을 포함하여 여러 가지의 암호화 작업을 선택 또는 결정해서 전송할 수 있도록 하는 보안 알고리즘을 개발이 중요한 과제이며, 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.

# 참고문헌


[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 Netwroks, 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] Prasanth Ganesan, Ramnath Venugopalan, Pushkin Peddabachagari, Alexander Dean, Frank Mueller, Mihail Sichitiu, "Analyzing and Modeling Encryption Overhead for Sensor Network Nodes", San Diego, California, WSNA'03, September 19, 2003.

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

[17] Mark Luk, Ghita Mezzour, Adrian Perrig, Virgil Gligor, "MiniSec: A Secure Sensor Network Communication Architecure", Proceedings of IEEE International Conference on Information Processing in Sensor Networks (IPSN), April, 2007.

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

[19] R. Rivest, "The MD5 Message-Digest Algorithm: RFC 1321", http://www.faqs.org/rfcs/rfc1321.html

[20] C. Adams, P. Cain, D. Pinkas, and R. Zuccherato, "Internet X.509 Public Key Infrastructure Time-Stamp Protocol (TSP): RFC 3161", http://www.ietf.org/rfc/rfc3161.txtAug , 2001.

[21] M. Myers, R. Ankney, A. Malpani, S. Galperin, and C. Adams, "X.509 Internet Public Key Infrastructure Online Certificate Status Protocol (OCSP): RFC 2560", http://www.ietf.org/rfc/rfc2560.txt, Jun. 1999.

[22] S. Kent and R. Atkinson, "Security Architecture for the Internet Protocol: RFC 2401", http://www.ietf.org/rfc/rfc2401.txt, Nov. 1998.

[23] A. Ferrari, F. Knabe, M. Humphrey, S. Chapin, and A. Grimshaw, "A flexible security system for metacomputing environments", in Proc. High Performance Computing and Networking Europe Amsterdam, The Netherlands, Apr. 1999.

[24] 홍도원,장구영,박태준,정교일, "유비쿼터스 환경을 위한 암호 기술 동향", 전자통신동향분석 제20권 제1호, 2005년

[25] L. Eschenauer and V. Gligor, "A Key-managementScheme for Distributed Sensor Networks", ACM CCS'02, pp.41-47, Nov. 2002

[26] M. Bellare, C. Namprempre, D. Pointcheval, M. Semanko, "The One-More_RSA-InversionProblems and the Security of Chaum's Blind Signature Scheme", Journal of Cryptology, pp.185-215, June 2003.

[27] 나재훈, 채기준, 정교일, "센서 네트워크 보안 연구 동향", http://kidbs.itfind.or.kr/WZIN/techtrnd/20-1_112_122.pdf

[28] L. Eschenauer and V. Gligor, "A Key-ManagementScheme for Distributed Sensor Network", Proc. of the 9th ACM Conference on Computer and Com-munications Security, 2002.

[29] D. Liu and P. Ning, "Establishing Pairwise Keys in Distributed Sensor Networks", Proc. of the 10thACM Conference on Computer and Communica-tion Security(CCS), 2003.

[30] D. Liu and P. Ning, "Location-Based Pairwise Key Establishments for Static Sensor Networks", Proc. of the 1st ACM Workshop on the Security of AdHoc and Sensor Networks(SASN), 2003.

[31] H. Chan, A. Perrig, and D. Song, "Random Key-Assignment for Secure Wireless Sensor Networks", Proc. of the 1st ACM Workshop on the Security ofAd Hoc and Sensor Networks(SASN), 2003.

[32] G. Jolly, Kuscu, and P. Kokate, "A Low-energy KeyManagement Protocol for Wireless Sensor Net-works", Proc. of the 8th IEEE International

Sympo-sium on Computers and Communications, June 2003.

[33] J. Deng, R. Han, and S. Mishra, "Enhancing BaseStation Security in Wireless Sensor Networks", Technical Report CU-CS-951-03, Department ofComputer Science, University of Colorado, Apr.2003.

[34] Yee Wei Law, Ricardo Corin, Sandro Etalle, andPieter H. Hartel, "A Formally Verified Decentral-ized Key Management Architecture for WirelessSensor Networks", Proc. of PWC'03, Sep. 2003.

[35] Malik Tubaishat, Jian Yin, Biswajit Panja, and San-jay Madria, "A Secure Hierarchical Model for Sen-sor Network", Proc. of SIGMOD, Mar. 2004.

[36] 서석충, 김형찬, R.S.Tamakrishna, "무선 센서 네트워크에서 타원 곡성 암호를 이용한 공유키 설정에 기반한 보안 프로토콜", 한국정보처리학회 추계학술발표대회 논문집 제12권 제2호, 2005.

[37] Chris Karlof, Naveen Sastry, David Wagner, "TinySec: A Link Layer Security Architecture for Wireless Sensor Networks", http://www.cs.berkeley.edu/~nks/papers/tinysec-sensys04.pdf

[38] M. Bellare, A. Desai, E. Jokipii, and P. Rogaway, "Aconcrete security treatment of symmetric encryption: Analysis of the DES modes of operation", InProceedings of 38th Annual Symposium on Foundations of Computer Science (FOCS 97), 1997.

[39] Samuel R. Madden, Robert Szewczyk, Michael J.Franklin, David Culler, "Supporting aggregatequeries over ad-hoc wireless sensor networks", InWorkshop on Mobile Computing and Systems Applications, 2002.

[40] 김대영, 김재언, 성종우, 이강우, "센서 네트워크 운영체제/미들웨어 기술동향", IITA 기술정책정보단, 한국 정보 통신 대학교 실시간 임베디드 시스템 연구실, 2005

[41] Ramnath Venugopalan, Prasanth Ganesan, PushkinPeddabachagari, Alexander Dean, Frank Mueller, Mihail Sichitiu, "Encryption overhead in embedded

systems and sensor network nodes: Modeling andanalysis", In 2003 International Conference onCompilers, Architectures and Synthesis for EmbeddedSystems, pages 188 - 197, 2003.

[42] 박승민, "센서 네트워크 노드 플랫폼 및 운영체제 기술 동향", 전자 통신 동향 분석 제 21권 제 1호, 2006년 2월.

[43] On World Report Wireless Sensor Networks : Mass Market Opportunities 2005, http://www.onworld.com/html/wirelesssensorsrprt.htm

[44] On World Report Wireless Sensor Networks : Growing Markets, Accelerating Demand 2005, http://www.onworld.com/html/wirelesssensorsrprt.htm

[45] Atmel Corporation, http://www.atmel.com

[46] ARM, http://www.arm.com

[47] 김원수, 장기수, "Zigbee 기술 동향 및 시장 전망 분석", 전자부품연구원 전자정보센터, 2003.

[48] Radio Pulse, http://www.radiopulse.co.kr

[49] Ason Hill, Robert Szewczyk, Alec Woo, Seth Hollar, David Culler, Kristofer Pister, "System Architecture Directions for Network Sensors". ASPLOS 2000, Cambridge, Nov, 2000.

[50] Ralph M. Kling, "Intel Mote: An Enhanced Sensor Network Node", Intel Workshop on Advanced Sensors, Structual Health Monitoring, and Smart Structures, Nov.2003.

[51] Intel Mote Homepage, http://www.intel.com/research/exploratory/motes.html

[52] Moteiv, http://www.moteiv.com/products/docs/an001-chan-ges.pdf

[53] Telos, http://www.moteiv.com/products/docs/an002-telos.pdf

[54] TinyOS Korea Community Forum, http://tinyos.keti.re.kr/

[55] RSA Conference, http://www.rsaconference.com/

[56] N. Koblitz, "Elliptic curve cryptosystems", in Mathematics of Computation 48, pp. 203 - 209, 1987.

[57] 이재용, "유비쿼터스 센서 네트워킹 기술", TTA저널 제95호 pp78 ~83, 2004.

[58] Trusted Computing Group, "Embedded Systems and Trusted Computing Security", http://www.trustedcomputinggroup.org/ 2005년

[59] 김복순, 조기환, 이행곤, "무선 센서 네트워크를 위한 계층적인 사전 키 분배 방법", 한국정보과학회 2004년도 봄 학술발표논문집 제31권 제1호(A), 2004.

[60] 나재훈, "USN(Ubiquitous Sensor Network) 인프라 보안 표준동향", 한국정보통신기술협회, 2004.

[61] 황성민, 김순자, "유비쿼터스 컴퓨팅 보안", 한국정보과학회지, 제21권 제5호, pp. 61~69, 2003.

[62] 박수용, 김성수, "센서 네트워크의 다층형 데이터 보안 방법", 한국정보과학회 2004년도 봄 학술발표논문집 제31권 제1호(A), 2004.

[63] 김대영, 도윤미, 박노성, 이상수, 팜민룡, 뒤뷔백, 파티오즈투르크, "센서 네트워크 기술", 정보처리학회지 10권 4호, 2003.

[64] Catharina Candolin, Hannu H. Kari, "A Security Architecture for Wireless AD Hoc Networks", In Proceedings of IEEE Milcom 2002, Anaheim, California, USA, October 2002.

[65] Jianwei Zhuge, Richard Yao, "Security Mechanisms for Wireless Home Network", Globecom 2003, IEEE Global Telecommunications Conference, 2003.

[66] 이상학, 김대환, 유준재, "유비쿼터스 센서네트워크 기술 개발 동향", 인터넷정보학회지 제5권 제3호, 2004.

[67] 안철현, "USN 동향과 USN 정보자원 관리체계", 한국전산원 차세대연구팀, 2007.

[68] 장선호, 이민경, 김재준, "유비쿼터스 센서 응용서비스 및 개발동향", IITA 기술정책정보단, 정보통신연구진흥원, 2006.

[69] Dimitrios N. Serpanos, Artemios G. Voyiatzis, "Increasing Symmetric Key Lifetime by Controlled Randomness", In "10th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA. 2005)". Facolta'

di Ingegneria, Catania, Italy, 2005.

[70] 윤정로, 최장욱, "유비쿼터스", 21세기북스, 2003.

[71] F. Stajano, "Security for Ubiquitous Computing", first Security& Privacry supplement to IEEE Computer, 4.2002.

[72] J. Zhu, S. Papavassiliou, S. Xu, "Modeling and Analyzing the Dynamics of Mobile Wireless Sensor Networking Infrastructure", Proceeding of IEEE Conference on Vehicular Technology Conference, pp.1550~1554, 2002.

[73] F. Stajano, R. Anderson, "The Resurrecting Duckling: Security Issues for Ubiquitous Computing", Wiley, 2002

[74] I.F. Akyildiz, S. Weilian, Y. Sankarasubramaniam, E. Cayirci, "A Survey on Sensor Networks", Communication Manazine IEEE, vol.40 no.8, pp. 102~114, 2002.

[75] 박승창, 남상엽, 류영달, 이기혁, 김완석, "유비쿼터스 센서 네트워크 기술", jinhan M&B, 2005.

[76] Alexander Joseph Huber, Josof Franz Huber, "유비쿼터스 모바일 컴퓨팅", jinhan M&B, 2004.

[77] 이정욱, "U-홈 네트워크", jinhan M&B, 2006.

[78] Haodong Wang, B.Sheng, Qun Li, "Elliptic curve cryptography-based access control in sensor networks", International Journal of Security and Networks, Vol. 1, No.3/4 pp. 127 - 137, 2006

[79] TinyECC : Elliptic Curve Cryptography for Sensor Networks, http:// discovery.csc.ncsu.edu/software/TinyECC/

[80] Lejla Batina, Nele Mentens, Kazuo Sakiyama, Bart Preneel, Ingrid Verbauwhede, "Low-Cost Elliptic Curve Cryptography for Wireless Sensor Networks", Security and Privacy in Ad-Hoc and Sensor Networks, Third European Workshop, ESAS 2006, Hamburg, Germany, September 20-21, 2006

[81] Leif Uhsadel, Axel Poschmann, Christof Paar, "An Efficient General Purpose

Elliptic Curve Cryptography Module for Ubiquitous Sensor Networks", Software Performance Enhancement for Encryption and Decryption (SPEED 2007), Amsterdam, NL Juni 11-12, 2007.

[82] Erik-Oliver Blaß, Martina Zitterbart, "Efficient Implementation of Elliptic Curve Cryptography for Wireless Sensor Networks", Technical Report, March 2005. ISSN 1613-849X

[83] Gaubatz, J.-P. Kaps, B. Sunar, "Public key cryptography in sensor networks - revisited", 1st European Workshop on Security in Ad-Hoc and Sensor Networks (ESAS 2004), Lecture Notes in Computer Science, volume 3313, Springer, Heidelberg, pages 2-18, August, 2004.

[84] Arvinderpal S. Wander, Nils Gura, Hans Eberle, Vipul Gupta, Sheueling Chang Shantz, "Energy Analysis of Public-Key Cryptography for Wireless Sensor Networks", Third IEEE International Conference on Pervasive Computing and Communication (PerCom 2005), Kauai, Mar. 2005.

[85] Wenliang Du, Jing Deng, Yunghsiang S. Han, Pramod K. Varshney, Jonathan Katz, Aram Khalili, "A pairwise key predistribution scheme for wireless sensor networks", Proc. 10th ACM Conf. Comp. and Commun. Security, pp 42-51, 2003.

[86] G. Murphy, A. Keeshan, R. Agarwal, E. Popovici, "Hardware-software implementation of public-key cryptography for wireless sensor networks", IET Irish Signals and Systems Conference (ISSC 2006)(CP519), p. 463 -468 , 2006.

[87] Ronald Watro, Derrick Kong, Sue-fen Cuti, Charles Gardiner, Charles Lynn, Peter Kruus, "Tinypk: securing sensor networks with public key technology", In SASN '04: Proceedings of the 2nd ACM workshop on Security of ad hoc and sensor networks, pages 59 - 64. ACM Press, 2004.

# ABSTRACT

## Light Weight Authentication Protocol based Security Framework for Ubiquitous Sensor Network

Zhung, Yonil
Ubiquitous Computing Lab
Department of Computer Engineering
Graduate School of Kyung Hee University
(Supervised by Professor Sungyoung Lee, Ph.D)

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. There are many researches to solve these issues, but many problems still exist. Also, current researches target specific sensor operating system, platform, or routing protocols to function properly. This is the reason why independent security policy is needed.

This paper proposes light weight security framework, independent to sensor OS, platform and routing protocols. The proposed framework supports security protocol for restricted sensor node environment in ubiquitous sensor network by using double-key based light weight security protocol and public-key based light weight security protocol. Also, it supports multiple sub-policies by using secure counter and transfer distance

control security protocols. These policies allow security manager to select and manage appropriate level of security.

The proposed framework supplements current security policy issues and power issues in the performance evaluations. It will be efficient to mobile sensor network, such as smart office and smart home.