유비쿼터스 센서 네트워크를 위한 공개키 기반의 보안 프로토콜 설계 및 구현

정연일*, 이승룡*
*경희대학교 컴퓨터공학과
e-mail: zhungs@oslab.khu.ac.kr

Design and Implementation of Public Key Based Light Weight Security Protocol for Ubiquitous Sensor Network

Yonil Zhung*, Sungyoung Lee*
*Dept of Computer Engineering, KyungHee University

요 약

유비쿼터스 컴퓨팅은 사용자에게 장소와 시간에 제약이 없이 자유롭게 네트워크에 접속 할 수 있는 환경을 제공하고 있다. 이러한 환경은 모든 정보의 공유 및 접근이 쉽게 이루어지는 반면, 인가되지 않은 사용자의 불법적인 접근도 쉽게 이루어질 수 있기 때문에 적합한 보안 정책이 필요하다. 특히 유비쿼터스 센서 네트워크의 센서 디바이스들은 제한된 전력을 이용하고 하드웨어적으로 작은 크기를 유지해야 하기 때문에 보안 정책 수립에 많은 제한이 발생하게 된다. 본 논문에서는 유비쿼터스 센서 네트워크에서 센서 노드의 제한적인 환경에서도 사용 가능한 공개키 기반의 보안 정책을 제안한다. 제안한 보안 프로토콜은 센서 노드에서 키 관리 및 보안 정책 적용을 최소한으로 포함시켜 저전력으로 사용 할 수 있는 공개키 기반의 보안 정책을 구현하였다. 성능 평가 결과 제안한 유비쿼터스 센서 네트워크에서 공개키를 사용한 저전력 보안 프로토콜은 센서 노드의 하드웨어 성능이 낮은 환경에서도 상대적으로 저전력으로 보안 정책의 활용이 가능 하였다.

1. 서론

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. 하지만 전력과 하드웨어의 제약을 고려하지 않은 기존의 보안 정책을 유비쿼터스 센서 네트워크에 적용하기에는 문제점이 발생한다. 따라서 유비쿼터스 센서 디바이스들의 제한된 전력과 하드웨어를 고려한 인증 프로토콜, 암호화 알고리즘과 같은 보안 정책 등이 필요하다. 일반적으로 센서 네트워크에서는 센서 노드의 제한적인 하드웨어 환경 때문에 암호키의 저장 및 관리의 어려움 많은 공개키 방식의 암호 정책 보다 대칭키 방식의 보안 정책이 대부분 제안되고 있다[2]. 하지만 최근 센서 하드웨어의 발전이 급속히 이루어지고 있으며 공개키 관리에 대한 문제점만 해결이 된다면 대칭키 기반의 암호 정책 보다는 공개키 기반의 암호 정책이 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 능력이 낮은 센서에서도 인증 및 암호화 알고리즘을 사용할 수 있도록 하는 공개키 기반의 보안 정책을 제안한다.

본 논문에서 제안하는 공개키 기반의 저전력 보안 프로토콜은 센서 노드에서는 키 관리의 부하가 거의 없고 공개키로 암호화 작업만 필요로 하기 때문에 센서 노드에서 다량의 키를 관리나 보관할 장소가 필요가 없다. 또한 제안하는 공개키 기반의 보안 프로토콜은 센서 노드에서 복호화 작업을 없애고 암호화 작업만 가능하게 해서 데이터를 전송하는 방식을 사용한다. 따라서 센서 노드의 하드웨어적 성능이 낮거나 저전력 상황에서도 인증 및 암호화 작업이 가능할 수 있게 하였다. 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 높은 레벨의 보안 정책을 펼칠 수 있게 한다.

2. 관련 연구

유비쿼터스 센서 네트워크는 가까운 미래에 널리 사용될 수 있는 기술로서 센서 네트워크 기반의 서비스에 대한 기술이 구체화 되면서 센서 네트워크상에서 보안에 대한 필요성이 점차 대두되어 보안 기술에 대한 연구가 점

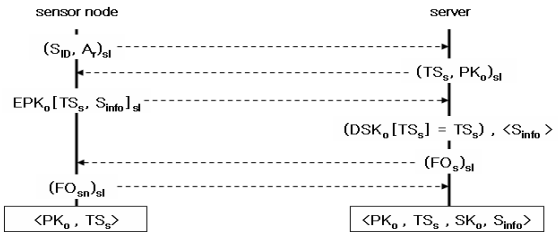
차 활발해지고 있다[1]. 키 관리 및 접근 제어 등의 암호 서비스를 제공하기 위한 연구로는 센서 노드용 운영체제로 개발된 TinyOS에서 사용 가능하도록 하는 TinySec이 있다[3]. TinySec에서는 링크 보안 기술을 적용한 간단한 접근제어와 메시지 위·변조를 막을 수 있는 무결성 기능과 비합법적인 노드에 의한 센서 정보의 해석을 방지하는 기밀성 기능을 제공한다. 또한, 산업체를 중심으로 하는 Zigbee Alliance는 저전력 무선 센서에 다양한 응용의 활용이 가능하도록 하는 기능과 보안 기능이 있다[4]. Zigbee 보안 서비스는 대칭키 암호 방식을 이용하여 두 노드 간의 비밀키 설정과 상호 인증 과정을 수행한다. 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. 센서 네트워크 보안 프로토콜로 초기에 발표된 SPINS(Security Protocols for Sensor Networks) 프로토콜은 멀티캐스팅 보안 프로토콜인 TESLA(Timed Efficient Stream Loss-tolerant Authentication)를 간소화시켜 개발한 μ-TESLA와 SNEP (Secure Network Encryption Protocol)로 구성된다. μ-TESLA는 다수로 구성된 센서 네트워크의 기기들의 인증을 담당하며 SNEP는 데이터의 기밀성, 인증을 보장한다[5][6]. 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.

3. 공개키 기반의 저전력 보안 프로토콜

유비쿼터스 센서 네트워크 환경에서 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 담당하게 된다. 센서 노드는 데이터를 전송하며 서버는 전송 된 데이터를 처리하는 작업만 담당하기 때문에 유비쿼터스 센서 네트워크 환경에서는 기존의 인증 방식과 다른 개념의 인증 방식 적용이 가능하다. 기존의 인증 방식은 양방향 인증으로 서로 상대 개체를 인증 하는 방식이지만 유비쿼터스 센서 네트워크 환경에서는 종속적인 인증이 가능하다. 이러한 환경이라면 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 없다. 또한 이러한 관계에선 데이터의 암호화 정책도 다른 방식으로 접근 할 필요가 있다. 센서 노드에서는 수집된 데이터를 암호화를 하는 알고리즘만 필요하며 복호화를 하는 알고리즘은 필요치 않게 된다. 서버와의 통신이 종속적이고 데이터의 흐름이 일방향이기 때문에 센서 노드에서는 복호화의 작업이 필요가 없게 된다. 따라서 센서 노드에서는 암호화에 필요한 공개키만 소유하고 있으면 되고, 센서 노드 자체에 복호화를 위한 키 혹은 인증은 필요 없게 된다. 하지만 이러한 환경에서 해결해야 할 사항은 센서 노드가 사용해야 할 공개키가 인가가 된 서버의 공개키를 소유해야 하는 과정이 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 논문에서는 센서 노드와 중앙처리 서버 간의 안전한 인증을 통한 공개키의 분배 와 관리 프로토콜을 제안한다. 유비쿼터스 센서 네트워크에서 공개키를 사용한 인증 프로토콜은 서버와 센서 노드 간의 인증 및 키 분배 프로토콜과 필요에 의한 공개키 변경 및 관리 프로토콜로 이루어져 있다.

3.1 공개키를 사용한 인증 프로토콜

유비쿼터스 센서 네트워크 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 인가된 서버에서 제공하는 공개키를 전송받고 수집 된 데이터를 암호화 하여 전송하는 작업을 담당하게 된다. 센서 노드가 인가된 서버에서 제공하는 공개키를 전송 받기 위해서는 공개키를 사용한 인증 프로토콜로 인가된 서버로부터 공개키를 전달 받고 서버는 새로운 센서 노드에 대한 정보와 해당 센서 노드의 공개키 관리를 하기 시작한다. (그림 1)에서는 센서 노드와 서버에서 공개키를 사용한 인증 프로토콜의 구조를 나타내었다.



(그림 1) 공개키 교환 인증 프로토콜

(그림 1)의 공개키 기반의 인증 프로토콜은 다음과 같은 순서로 동작이 된다. 센서 노드에서 센서 노드가 동작을 시작하게 되면 먼저 서버와 인증 작업을 하기 위해서 신호를 보내게 된다($(S_{ID},Ar)sl$). 센서 노드는 서버와의 인증 작업이 끝나기 전까지 어떠한 작업도 하지 않는다. 서버에

서는 해당 센서 노드의 정보를 받은 후에 공개키와 비밀키를 생성 하고 타임스탬프를 생성해서 센서 노드로 보내게 된다((TS_s,PK_o)sl). 서버에서는 해당 센서 정보와 연관있는 비밀키, 공개키, 타임스탬프를 임시 보관한다. 공격자에 의해서 인증 요청이 오더라도 공개키와 타임스탬프는 공개가 되어도 의미가 없기 때문에 문제가 없다.

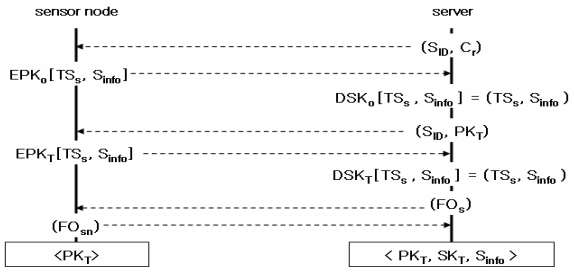
<표 1>프로토콜에서 사용하는 인자

S_{ID}	센서 ID
Ar	센서 노드에서 보내는 인증 요청 신호
( )sl	단거리 RF 전송
S_{info}	센서의 고유 정보
Cr	서버에서 보내는 공개키 변경 신호
TS_s	서버에서 생성한 타임스탬프
PK_o, SK_o	서버에서 생성한 공개키, 비밀키
PK_T, SK_T	서버에서 생성한 새로운 공개키, 비밀키
$EPK_0(DSK_0)$	공개키(비밀키)를 이용하여 암호화(복호화)
FOs	서버에서 보내는 종료 확인 신호
FOsn	센서 노드에서 보내는 종료 확인 신호
< >	각 객체에서 저장해야 하는 값

센서 노드에서는 받은 공개키를 이용해서 서버로부터 받은 타임스탬프와 센서 노드 자신의 정보를 암호화 하여 전송 한다($EPK_0[TSs,S_{info}]$sl). 공개키를 이용하여 암호화 하였기 때문에 공개키를 전송 한 서버 이외는 복호화가 불가능하게 된다. 서버에서는 전송 받은 데이터를 서버에서 보관 중인 비밀키로 복호화를 한다. 복호화를 해서 나온 데이터와 자신이 가지고 있는 타임스탬프와 비교를 함으로 현재 인증을 받으려는 센서 노드로부터 왔는지를 확인을 한다($DSK_0[TSs]$=TSs). 확인이 끝나고 나서 센서 노드로 인증 종료 신호를 보낸다((FOs)sl). 센서 노드에서도 서버로부터 종료 신호를 받은 후 종료 서버로 종료 확인 신호를 보낸다((FOsn)sl). 이러한 작업이 종료가 되기 전 다른 상황이 발생하면 진행 중인 과정은 취소가 된다. 모든 작업이 끝나고 종료 신호까지 전송을 하면 센서 노드에서는 서버의 공개키와 타임스탬프를 저장한다. 서버에서는 비밀키, 공개키, 타임스탬프, 그리고 해당 센서의 정보를 저장한다.

3.2 키 교체 및 관리 프로토콜

공개키 기반의 암호 정책을 사용하더라도 키의 크기와 키 사용 기간에 따라 공개키를 교체하는 것이 네트워크 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. 양쪽의 관계가 종속 적인 상태에서 공개키 분배는 가장 중요한 부분이다. 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 2) 서버와 센서 노드간의 키 교체 알고리즘

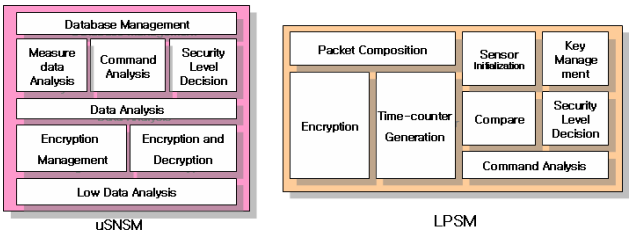
서버에서는 공개키 교체 주기적으로 또는 유비쿼터스 센서 네트워크 보안 관리자의 판단에 의해 공개키 교체 신호를 센서 노드로 전송하게 된다(S_{ID},Cr). 센서 노드에서는 서버의 교체 신호가 오면 저장해 두었던 타임스탬프와 센서 정보를 기존의 공개키로 서버로 전송을 하게 된다($EPK_0[TSs,S_{info}]$). 서버에서는 전손된 데이터를 기존의 비밀키로 복호화를 하고 타임스탬프 값과 센서 정보 값이 기존에 저장되어 있던 값과 일치하는지 검사를 한다($DSK_0[TSs,S_{info}]$=(TSs,S_{info})). 만일 서버에서 키 교체 요청을 하지 않았는데도 해당 값이 전달되었다면 공격자의 공격이라고 판단하게 된다. 또한 공개키 교체 신호를 전송했더라도 응답이 없으면 공격자의 공격이라고 판단 한다. 서버에서 데이터 값이 일치를 하면 새로운 공개키를 생성하여 전달하게 된다(S_{ID},PK_T). 전손 되는 값은 해당 센서의 ID와 공개키이기 때문에 값이 복사가 되더라도 문제가 없게 된다. 센서 노드에서는 공개키 교체 신호를 받고 응답을 했지만 새로운 키를 전송 받지 못하면 공격자의 공격으로 판단한다. 새로운 공개키를 받은 센서 노드에서 새로운 공개키를 이용해서 저장된 타임스탬프와 센서 정보를 암호화를 해서 다시 서버에 응답을 한다($ESK_T[TSs,S_{info}]$). 공격자의 공격에 의해서 다른 공개키를 이용해서 공격자에게 데이터를 보내더라도 공격자의 공격 여부만 판단하면 신뢰 받은 서버와 센서 노드간의 타임스탬프와 센서 정보는 수정이 가능하기 때문에 공격자에게는 의미 없는 정보가 된다. 서버에서는 센서에 분배된 키가 안전하게 전송 되었는지 확인하기 위해서 센서 노드에서 전송한 데이터를 새로운 비밀키를 이용해서 복호화를 한 뒤 확인을 한다($DSK_T[TSs,S_{info}]$=(TSs,S_{info})). 값이 틀리거나 복호화가 불가능 할 경우 공격자의 공격으로 판단하게 된다. 모든 검사가 끝나고 나면 서버에서 센서 노드로 키 교체 알고리즘 종료신호를 보내고(FOs), 센서 노드에서도 서버로 키 교체 완료 신호를 보내게 된다(FOsn). 이 작업이 끝나면 각각의 센서 노드와 서버에서는 새로운 공개키, 비밀키를 저장하게 된다.

기존의 공개키 기반의 보안 정책은 암호화를 하는 키를 공개 하는 방식이기 때문에 신뢰가 있는 인증 서버를 이

용하여 키 관리 및 분배를 하지만 유비쿼터스 센서 네트워크와 같이 종속족인 관계의 양쪽 노드에서 공개키를 이용한 보안 정책을 가질 경우 인증 및 공개키 분배가 어려운 문제로 부각된다. 본 논문에서 제안된 프로토콜의 경우 데이터 전송 및 수신 시간제한 및 데이터의 비교 등으로 키 교체 중간에 공격자의 공격이라고 판단되면 모든 과정은 취소가 되고 중앙 관리 서버에서 네트워크를 감시하게 되므로 안전한 공개키 분배라고 할 수 있다.

4. 구현 및 성능 평가

서버에는 유비쿼터스 센서 네트워크 보안 관리자(uSNSM)를 설계하여 인증, 키 교환 등의 암호 정책을 모듈별로 처리를 할 수 있게 하였으며, 센서 노드에는 저전력 보안 관리자(LPSM)를 기존의 센서 노드에 포함을 시켜 보안 프로토콜을 처리 할 수 있도록 하였다(그림 3).



(그림 3)서버와 센서의 보안 관리자 모듈 구조

저전력 보안 관리자가 동작되는 센서 노드의 하드웨어 사양은 ATmega128L, 512KB 플래쉬메모리, RF CC2420 (2.4GHz, IEEE802.15.4, 250kbps Effective Data rate), 운영체제로는 Qplus를 사용하였다. 그리고 센서 노드의 보안 관리자는 C를 이용하여 프로그래밍 하였으며 서버의 데이터 처리는 Windows XP 환경에서 C++를 이용하여 프로그래밍 하였다. 보안 정책이 적용되기 전의 센서 노드에서 보내는 패킷의 크기는 라우팅 헤더 부분을 제외하고 7byte였으며 보안 정책이 포함 되고 나서 패킷은 보안 데이터를 포함하여 10byte로 3byte가 증가 되었다. 그 이외 인증 및 공개키 교체를 위한 데이터 패킷으로 10byte, 17byte 패킷을 추가로 정의해서 사용하였다. 센서 노드에 포함되는 코드의 경우에는 기본의 공개키 보안 모듈을 모두 포함한다면 컴파일 하기전의 코드 크기는 71828 byte 이상 포함이 되어야 했으며 본 논문에서 제안한 공개키 보안 프로토콜은 50989 byte가 포함되어 활용 가능 하였다. 실제 센서 노드에서 전송되는 데이터 중에 센서 노드에서 수집한 값과 보안 관련 데이터만 암호화를 하여 전송하기 때문에 암호화 데이터는 7byte에서 14byte만 처리하도록 하였으며 인증과 키 교체 시에만 보안 관련 연산을 추가적으로 하도록 설계하였다. 공개키를 사용하는 보안 정책임에도 불구하고 공개키 정책의 핵심적인 키 생성, 관리, 저장 등의 보안 연산은 제외가 되어서 센서 노드의 연산량을 줄였지만 공개키 기반의 보안정책은 유지 할 수 있었다.

5. 결론 및 향후 연구

유비쿼터스 컴퓨팅 발전에 가장 중요한 역할을 하는 무선 센서 네트워크는 기존의 보안 정책을 적용하기가 힘든 분야이다. 본 논문은 유비쿼터스 센서 네트워크 보안의 무선을 이용하기 때문에 생기는 무선 보안 문제와 센서 디바이스의 제한된 환경에 필요한 보안 정책 가운데 센서 노드의 제한된 환경으로 생기는 보안의 문제점에 대해서 논의하고 그 해결 방법을 제시하였다.

본 논문의 공개키 기반 저전력 보안 프레임워크는 서버와 센서 노드간의 종속적인 관계를 이용하여 인증 및 키 교체 알고리즘을 제안 하였다. 대부분의 보안 관련 연산은 서버에서 처리하도록 하고 센서 노드에는 적은 연산만 처리 할 수 있도록 설계하여 저전력으로 안전한 유비쿼터스 센서 네트워크를 구성할 수 있도록 하였다. 또한 타임스탬프를 이용한 보안과 전송 거리 조절을 이용한 보안 정책을 이용하여 공격자의 공격에 능동적으로 대처를 할 수 있게 지원하여 네트워크의 안정성을 지원하였다. 본 논문은 유비쿼터스 센서 네트워크에서 센서의 운영체제와 플랫폼, 데이터 전송 라우팅 방식에 종속적이지 않은 보안 프로토콜이기 때문에 다른 사양의 센서 노드에서 사용 가능한 확장성을 가지고 있다. 성능 평가 결과 저전력으로 안전한 네트워크 상태를 지원 할 수 있어서 스마트 오피스 및 스마트 홈과 같은 실제 환경의 유비쿼터스 센서 네트워크에 적용하는 보안 정책으로 적합하다고 할 수 있다. 향후 연구 과제로 유비쿼터스 센서 네트워크 상태를 모니터링 하여 현 센서 네트워크 상태에 맞는 보안의 적정한 레벨을 선택하고 관리할 수 있도록 하는 모니터링 서비스를 지원하며, 다른 레벨의 보안 알고리즘을 네트워크의 상태 정보에 맞게 교체 하는 보안 프레임워크를 개발하는 것도 앞으로의 주요 연구 과제이다.

참고문헌

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

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

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

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

[5] Adrian Perrig et al."SPINS:Security Protocols for Sensor Networks", Proceedings of Seventh Annal International Conference on Mobile Computing and Networks, July 2001.

[6] Adrian Perrig et al."Efficent Authentication and Signing of Multicast Streams over Lossy Channels" IEEE Symposium on Security and Privacy, May 2000.