SWATT 기법을 이용한 센서 노드 간 상호 검증 프로토콜

허 경 수[†] · 최 현 우[††] · 장 현 수[†††] · 엄 영 익[††††]

요 약

센서 노드의 프로그램 변경을 통한 센서 네트워크 공격을 방지하는 것은 중요한 센서 네트워크 관련 보안요소들 중 하나이다. 이를 해결하기 위해 각 센서 노드 내에서 실행 중인 코드가 변경되지 않았음을 검증하는 소프트웨어 기반 검증(attestation) 기술이 사용되고 있다. 그러나 현재 제안되고 있는 소프트웨어 기반 검증 기법은 신뢰할 수 있는 검증자가 존재하지 않는 센서 네트워크 환경에 대한 고려가 부족할 뿐만 아니라, 망의 변화가 없는 정적인 환경에 중점을 두어 설계되었기 때문에 센서 네트워크 환경에 적합하지 않다. 본 논문에서는 SWATT(Software-based ATTestation) 기법을 이용하여 센서 네트워크 환경에 적합한 상호 검증 프로토콜을 제안한다. 제안 기법은 신뢰 노드가 존재하지 않는 센서 네트워크 환경에서 이웃한 센서 노드에게 자신의 존재를 주기적으로 알리되, SWATT 기법을 이용하여 이를 인지한 이웃 센서 노드와 상호 검증을 수행하도록 한다. 이를 통해 신뢰할 수 있는 검증자가 없는 센서 네트워크 환경에서 악의적인 공격으로 인해 변경된 프로그램을 실행하는 센서 노드를 식별할 수 있으며, 결과적으로 검증된 센서 노드들만으로 이루어지는 네트워크 구성이 가능하도록 한다.

키워드 : 센서 네트워크, 센서 네트워크 보안, 상호 검증, SWATT(Software- based ATTestation)

Mutual Attestation Protocol using Software-based Attestation Scheme in Sensor Network Environments

Kyungsoo Heo[†] · Hyunwoo Choi[††] · Hyun-Su Jang[†††] · Young Ik Eom[††††]

ABSTRACT

Prevention of attacks being made through program modification in sensor nodes is one of the important security issues. The software-based attestation technology that verifies the running code by checking whether it is modified or not in sensor nodes is being used to solve the attack problem. However, the current software-based attestation techniques are not appropriate in sensor networks because not only they are targeting static networks that member nodes does not move, but also they lacks consideration on the environment that the trusted verifier may not exist. This paper proposes a mutual attestation protocol that is suitable for sensor networks by using SWATT(Software-based ATTestation) technique. In the proposed protocol, sensor nodes periodically notify its membership to neighbor nodes and carry out mutual attestation procedure with neighbor nodes by using SWATT technique. With the proposed protocol, verification device detects the sensor nodes compromised by malicious attacks in the sensor network environments without trusted verifier and the sensor networks can be composed of only the verified nodes.

Key Words : Sensor network, Sensor network security, Mutual Attestation, SWATT(Software-based ATTestation)

1. 서 론

센서 네트워크는 안전 감시, 빌딩 보안, 트래픽 흐름 측정, 환경오염 추적, 군사 보안 등 다양한 응용 분야에 적용할 수 있다. 또한 가까운 미래의 유비쿼터스 컴퓨팅 환경에서의 서비스 제공을 위한 다양한 정보의 수집을 위해 센서 네트워크 기술이 사용될 수 있다. 센서 네트워크를 구성하는 다수의 센서 노드는 공격자에 의해 물리적으로 쉽게 획득되어 악의적인 변경을 통한 공격이 가능한 보안상의 취약점이 존재한다. 따라서 센서 네트워크 설계 시 공격자에 의한 센서 노드의 소프트웨어적/하드웨어적인 변경을 차단하거나 감지할 수 있는 방법이 고려되어야 한다. 센서 노드에 대한 소프트웨어적/하드웨어적인 공격 중 센서 노드의 프로그램 변경을 통한 센서 노드의 손상과 센서 네트워크에 대한 악의적인 공격은 센서 네트워크에 관련된 다양한 보안 위협 중 가장 해결하기 어려운 과제 중 하나이다[1]. 그러나

† 준 회 원 : 성균관대학교 이동통신공학과 석사과정
†† 준 회 원 : 성균관대학교 전기전자컴퓨터공학부 석사과정
††† 준 회 원 : 성균관대학교 전기전자및컴퓨터공학과 박사과정
†††† 종신회원 : 성균관대학교 정보통신공학부 교수
논문접수 : 2007년 11월 28일, 심사완료 : 2008년 1월 3일

DOI: 10.3745/KIPSTC.2008.15-C.1.9

각 센서 노드의 비용을 낮추어야 하는 센서 네트워크의 특징으로 인해 하드웨어적인 장치를 탑재하여 센서 노드의 프로그램 변경 공격에 대응하기는 어렵다. 이러한 단점을 극복하고 센서 노드에 대한 프로그램 변경 문제를 해결하기 위해 각 센서 노드에서 실행중인 코드가 변경되지 않았음을 검증하는 여러 검증 기술을 사용할 수 있다. 악의적인 목적으로 변경된 프로그램이 실행중인 센서 노드의 메모리 영역은 변경되지 않은 정상적인 프로그램이 실행중인 센서 노드의 메모리 영역의 상태와 다르기 때문에 센서 노드의 프로그램 메모리 영역을 검증함으로써 악의적인 공격이나 예상치 못한 오류로 인해 변경된 프로그램이 실행중인 센서 노드를 탐지할 수 있다.

코드 검증(attestation)은 소프트웨어적/하드웨어적인 기법을 통해 수행할 수 있다. 하드웨어적인 검증은 TCG(Trusted Computing Group)[2][3], NGSCB(Next-Generation Secure Computing Base)[4]에서 개발된 하드웨어를 탑재한 장비를 통해 이루어지며 소프트웨어적인 검증은 검증자와 센서 노드 간 통신을 통해 센서 노드의 메모리 영역에 대한 검증을 수행함으로써 이루어진다. 센서 노드는 낮은 비용을 요구하기 때문에 현재 하드웨어적인 검증 기법 보다는 소프트웨어를 통해 검증을 수행하는 기법이 주로 제안되고 있다. 기본 임계값 비밀 공유 방식과 과반수 투표에 기반 한 검증 방식[5]은 소프트웨어적 검증 기법으로 사용되었지만 네트워크의 망 변화가 발생하지 않는 정적인 네트워크 상태를 기반으로 하고 있으므로 현재의 센서 네트워크 환경에는 적합하지 않다. 또 다른 소프트웨어적 검증 기법으로 challenge-response 프로토콜을 사용하여 검증 장비와 대상 디바이스 사이의 검증을 수행하는 SWATT(SoftWare-based ATTestation)[6] 기법은 임베디드 디바이스의 검증에 효과적으로 사용될 수 있지만, 센서 네트워크의 여러 특징으로 인해 이를 그대로 센서 네트워크 환경에 적용할 수 없다. 본 논문에서는 소프트웨어적 디바이스 검증 기법인 SWATT 기법을 이용하여 센서 네트워크 환경에 적합한 센서 노드 메모리 검증 프로토콜을 제안한다. 즉, 제안하는 프로토콜은 신뢰할 수 있는 센서 노드가 존재하지 않는 센서 네트워크 환경에서 이웃한 센서 노드에게 자신의 존재를 주기적으로 알리고 이를 인지한 센서 노드와 SWATT를 이용하여 상호 검증을 수행한다. 이때 센서 노드는 상호 검증된 노드 리스트를 생성하고 리스트에 해당 센서 노드가 존재하지 않을 경우에만 상호 검증을 수행한다. 이를 통해 신뢰할 수 있는 검증자가 없는 센서 네트워크에서 공격으로 인해 손상된 센서 노드를 식별할 수 있으며 검증된 센서 노드만으로 안전한 센서 네트워크를 구성할 수 있다.

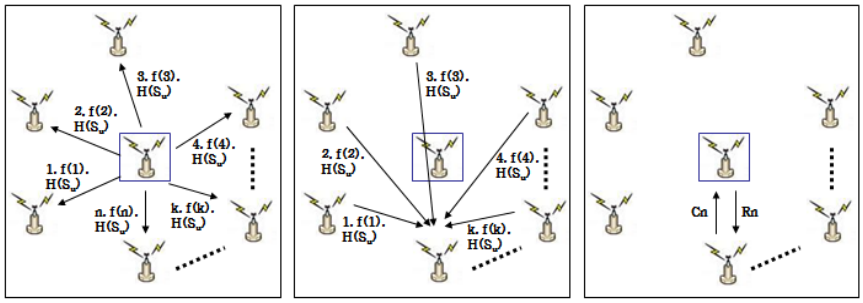
본 논문은 다음과 같이 구성되어 있다. 2장에서는 하드웨어적인 검증 기법과 소프트웨어적인 검증 기법의 특징에 대해 구체적으로 설명한다. 3장에서는 제안 프로토콜을 위한 가정사항 및 위협모델을 제시하고 센서 네트워크 환경에 적합한 센서 노드 메모리 검증 프로토콜의 설계를 위한 요구사항을 도출한다. 그리고 제안 프로토콜인 SWATT(SoftWare-based ATTestation)을 이용한 센서 노드 간 상호 검증 프로토콜에 대해 구체적으로 설명한다. 4장에서 제안 프로토콜의 보안 검증 및 성능 분석을 수행하여 제안 기법의 효용성을 입증하고 마지막으로 5장을 통해 논문을 요약하고 결론 맺는다.

2. 검증(Attestation) 기술

검증(attestation)이란 검증 대상 디바이스의 소프트웨어적/하드웨어적 변경이 공격자에 의해 임의로 이루어지지 않았음을 증명하는 것으로, 즉 해당 디바이스의 소프트웨어적/하드웨어적인 무결성을 증명하는 과정이다. 디바이스 간 검증 기법에는 하드웨어 기반의 검증 기법과 소프트웨어 기반의 검증 기법이 있다. 본 장에서는 디바이스 간 검증을 위해 개발된 대표적인 소프트웨어/하드웨어 기반 기법에 대해 설명하고, 센서 네트워크 환경에 응용 가능한 소프트웨어 기반 기법인 SWATT의 특징을 기술한다.

2.1 하드웨어 기반 검증 기법

TPM(Trusted Platform Module)[2][3]은 TCG(Trusted Computing Group)에서 규정한 하드웨어 기반 attestation 방식으로 신임 성립을 위해 신뢰 장치로써 스마트카드를 사용한다. TPM은 임의 수 생성, 키 계산, 해시 계산을 위한 향상된 하드웨어 계산 엔진과 암호화키를 위한 보호된 저장 영역을 제공한다. TPM은 대칭키/비대칭키를 생성하고 저장할 수 있으며 비대칭 암호화 연산을 수행할 수 있다. 또한 TPM은 플랫폼 의존적인 설정 값을 저장하기 위해 사용되는 PCRs(Platform Configuration Registers)을 제공한다. PCRs은 전원 인가 시 초기화되고 소프트웨어 무결성 값을 저장하기 위해 사용된다. 소프트웨어 컴포넌트(BIOS, bootloader, OS, applications)는 실행되기 전에 TPM으로 평가되어 계산된 해시 값이 특정한 PCR에 저장된다. BIOS에 상주하고 전원 인가 시 처음으로 실행되는 CRTM(Core Root of Trust Measurement)이 자기 자신과 BIOS를 평가하고 다음 소프트웨어 컴포넌트로 컨트롤을 넘긴다. 모든 평가된 컴포넌트는 SML(Stored Measurement Log)을 생성하고 해당 결과를 기록한다. PCR값은 원격에 있는 장비가 해당 장비의 상태를 attestation하기 위해 SML과 함께 사용할 수 있으며 PCR값의 정확성을 보장하기 위해 이동 불가능한 키 값인 AIK(Attestation Identity Key)를 통한 서명을 이용한다. 원격에 있는 장비는 서명 검증을 통하여 해당 장비의 무결성을 확인하고 PCR값을 비교할 수 있다. 그리고 TPM은 데이터 블록을 특정 플랫폼 설정 값으로 연결할 수 있는 sealing이라는 기능을 제공한다. 봉인된 메시지는 플랫폼 설정 레지스터의 범위 값, 이동 불가능한 키 값과 봉인된 데이터 블록으로 구성되어 있다. TPM은 sealing이 실행된 플랫폼 설정 값과 현재의 플랫폼 설정 값이 동일하다면 봉인된 메시지를 복호화하여 봉인된 데이터 블록을 전송할 수 있게 된다. 그러나 TPM 칩을 이용한 방식은 개별 센서 노드의 구성비용이 낮은 센서 네트워크에는 적합하지 않다.



(그림 1) 기본 임계값 비밀 공유 방식의 검증 단계[5]

2.2 소프트웨어 기반 검증 기법

대표적인 소프트웨어 기반 검증 기법으로 기본 임계값 비밀 공유 방식과 과반수 투표에 기반 한 검증 방식, 그리고 SWATT(SoftWare-based ATTestation) 기법이 있다.

2.2.1 기본 임계값 비밀 공유 방식과 과반수 투표에 기반 한 검증 방식

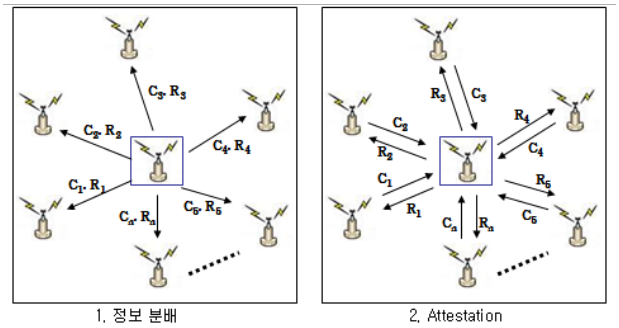
기본 임계값 비밀 공유와 과반수 투표에 기반 한 attestation 기법에서는 센서 노드의 배치 전에 빈 프로그램 메모리 영역에 noise을 저장하며 체크섬 계산 시 블록 기반 pseudo 임의 메모리 탐색을 사용한다. Noise 생성은 0부터 시작되는 카운트 값을 각 노드마다 고유한 seed 값을 사용하여 RC5로 암호화한다. 블록 기반 pseudo 임의 메모리 탐색 방식은 이어서 설명할 SWATT 기법처럼 한 번에 하나의 메모리 셀의 값을 가져오는 것이 아니라 한 번에 블록 단위로 메모리의 값을 가져와서 체크섬 계산을 수행하는 방식이다[5].

기본 임계값 비밀 공유 방식은 다음의 3단계로 구성되어 있다.

1. 센서 노드는 배치 전에 고유한 noise를 생성하여 빈 메모리 영역에 저장한다. 센서 노드의 배치 후, 최소한 센서 노드가 공격자에 의해 변경되기 전에 센서 노드는 이웃한 센서 노드들을 발견하고 발견된 센서 노드의 수보다 작은 임의의 상수를 생성하여 다음 다항식(1)을 생성한다. 다항식 (1)을 사용하여 1부터 이웃한 노드의 수까지 $x, f(x)$를 구한 후 해시된 S_U와 함께 이웃한 센서 노드들에게 분배한다.

$$f(x) = S_U + a_1 x^1 + a_2 x^2 + \cdots, + a_{k-1} x^{k-1}$$
$$(S_U : noise\ 생성\ seed, k : 1 \le k \le \text{이웃한노드수}) \quad (1)$$

2. 검증을 수행할 센서 노드의 이웃 센서 노드 중에서 특정 클러스터 헤더 선출 알고리즘을 기준으로 헤더를 선출한 후, 이웃한 센서 노드로부터 이전에 분배되었던 n개의 $(x, f(x), H(S_U))$ 중에서 k개를 수신하여 S_U를 계산한다.
3. 헤더는 검증을 수행할 센서 노드에게 임의의 challenge값을 전송하고 자체적으로 계산된 S_U를 통해 해당 노드의 체크섬을 내부적으로 계산한다. 최종적으로 검증을 수행할 센서 노드로부터 수신된 체크섬과 내부적으로 계산된 체크섬이 동일하면 해당 노드는 공격자에 의해 변경되지 않은 정상적인 센서 노드로 판단된다.



(그림 2) 과반수 투표에 기반 한 검증 방식의 검증 단계

기본 임계값 비밀 공유 방식의 검증 기법은 클러스터 헤더로 선출된 센서 노드가 공격자에 의해 변경되었다면 검증의 정확성을 입증할 수 없는 문제가 있다. 따라서 특정한 클러스터 헤더 결정에 의한 방식이 아닌 검증할 노드의 이웃 노드들에 의한 과반수 투표에 기반 한 검증 기법이 개발되었다. 과반수 투표에 기반 한 검증 기법은 2단계로 구성되어 있다.

1. 센서 노드의 배치 전에 고유한 seed를 통한 noise와 배치 시 예상되는 n개의 challenge/response 쌍을 미리 계산하여 메모리에 저장. 센서 노드 배치 후 n개의 challenge/response 중 하나씩을 임의로 선택하여 발견된 이웃 센서 노드에 전송한다.
2. Challenge/response를 수신한 이웃 센서 노드는 이를 전송한 센서 노드와 검증을 수행할 때 수신된 challenge와 함께 replay 공격을 방지하기 위한 일련번호를 해당 센서 노드에게 전송한다. Challenge와 일련번호를 수신한 센서 노드는 메모리 체크섬을 계산하여 response를 전송하게 된다. 최종적으로 이웃한 센서 노드가 전송한 challenge에 대한 response와 이전에 수신한 response가 과반 수 이상 동일하다고 인정되면 해당 노드는 정상적인 센서 노드로 판단된다[5].

앞서 설명된 두 방식은 검증할 노드의 다수개의 이웃한 노드들이 서로 협력하는 방식, 즉 센서 노드의 배치 시 이웃한 노드에게 검증에 필요한 데이터를 분배하는 방식을 사용하기 때문에 분배된 데이터를 가진 이웃한 노드가 사라진 경우 검증을 수행할 수 없는 경우가 발생하므로 네트워크 망의 변화가 발생하는 센서 네트워크 환경에는 부적합하다. 또한 두 방식은 이웃한 노드들이 많은 경우 분배된 정보의 저장을 위해 많은 메모리 공간을 요구하고 있으며 과반수 투표에 기반 한 검증 방식은 1개 센서 노드의 검증을 위해 많은 이웃한 노드들이 검증을 수행해야 하는 부담이 존재한다. 그리고 검증의 수행 후 공격받아 손상된 노드는 식별할 수 없는 문제가 있다.

2.2.2 SWATT(Software-based ATTestation) 기법

SWATT 기법은 하드웨어 기반 검증 기법과는 다르게 TPM 칩과 같은 추가적인 하드웨어 장비 없이 소프트웨어적으로 임베디드 디바이스의 메모리 내용(code, static data, 디바이스 설정 값 등)을 검증하기 위해 개발된 기술[2][6]로서 임베디드 디바이스뿐만 아니라 다양한 컴퓨팅 시스템의 검증에 사용되고 있다[7]. SWATT 기법은 원격에 디바이스를 검증할 수 있는 검증장비를 두고 임베디드 디바이스 간에 challenge-response 프로토콜을 사용하여 검증을 수행한다. SWATT 기법은 추가적인 하드웨어 비용이 필요하지 않기 때문에 수백 또는 수천 개의 센서 노드로 구성되는 센서 네트워크 환경에 적합하다.

SWATT 방식에서 검증장비는 미검증된 디바이스에 임의의 challenge를 보낸다. Challenge를 수신한 디바이스는 메모리의 체크섬을 계산하여 검증장비에 response를 반환한다. 이때 검증장비는 디바이스에 내장된 메모리 값을 알고 있어 디바이스의 메모리 체크섬을 내부적으로 계산하여 계산된 메모리의 체크섬과 반환된 체크섬을 비교하여 공격에 의한 디바이스의 변경 여부를 확인하고 정상적인 디바이스로 검증되면 이를 인증한다. (그림 3)은 SWATT 기법에 의한 디바이스 검증 방법을 간략히 보여준다.

이 경우 공격자는 디바이스의 빈 메모리 영역에 원래 디바이스의 메모리 값을 저장하여 체크섬 계산 시 사용하여 정확한 결과 값을 전송할 수 있다. 따라서 SWATT에서는 challenge와 response간 예상되는 계산시간 이상이 걸린다면 반환 값이 정확하더라도 해당 디바이스의 메모리는 공격자에 의해 변경된 것으로 간주한다. 즉, 변경된 메모리를 감추기 위해 변경된 메모리 검증(체크섬) 함수에서 또 다른 일을 해야 하기 때문에 체크섬 계산시간이 증가하여 이를 통해 간접적으로 메모리의 변경을 감지할 수 있다.

SWATT는 공격자가 디바이스의 메모리값을 모두 알 수 있다고 가정하나 디바이스의 하드웨어(메모리의 크기, 메모리 접근 시간 변경, 프로세서 처리 속도 변경)는 변경하지 않는다는 것을 전제로 하고 있다. 또한 검증장비는 디바이스의 하드웨어 정보를 알고 있다고 가정한다.

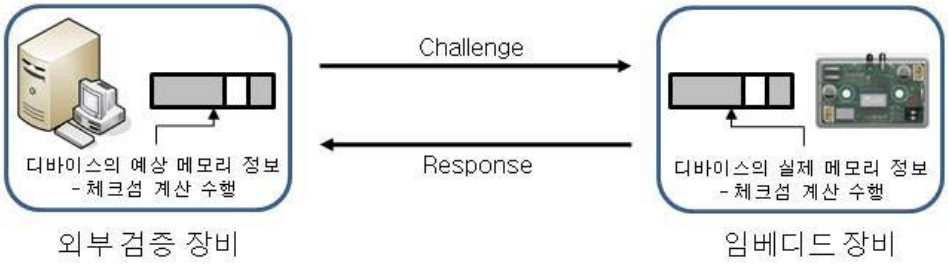
SWATT의 특징은 아래와 같다.

- RC4 Pseudo-random generator(PRG)을 사용하여 메모리 접근을 위한 주소를 생성하여 공격자가 메모리 변경 시 변경된 메모리로의 접근인지를 체크하게 하여 계산시간을 증가시킨다.
- Pseudo-random generator(PRG)의 seed로 사용되는 임의의 challenge를 통하여 재연과 선 계산 공격에 강하다.
- 비록 임의의 순서로 메모리를 접근하지만 $O(n\ln n)$의 메모리 접근을 통하여 모든 메모리를 한번은 접근하도록 하여 작은 메모리 변경도 감지할 수 있다.
- 디바이스에 최적화된 검증 함수를 통해 공격자가 더 이상 검증 함수를 최적화할 수 없게 하여 예상되는 시간 안에 체크섬 계산을 수행할 수 없게 한다.
- 체크섬 계산 시 접근되는 메모리 주소의 생성에 이전 체크섬을 사용하여 병렬 계산을 수행하지 못하게 한다.

그러나 SWATT는 신뢰받은 검증자가 존재하는 경우에 유용한 방식으로 신뢰받은 검증자가 존재하지 않는 센서 네트워크에는 부적합하며 1hop으로 모든 센서 노드를 검증할 수도 없다. 또한 검증을 통해 손상된 노드로 판별된 센서는 물리적인 제거를 통해 네트워크에서 배제되어야 하며 검증의 수행 후 공격받아 손상된 노드는 식별할 수 없는 문제가 있다.

3. SWATT 기반 센서 노드 간 상호 검증 프로토콜

본 장에서는 센서 네트워크 환경에서 효과적으로 센서 노드 간 메모리 검증을 수행하는 프로토콜을 설계하기 위한 주요 가정 사항 및 센서 네트워크 환경에서의 위협 모델을



(그림 3) SWATT의 challenge-response 프로토콜

제시한다. 이로부터 제안 프로토콜 구성을 위한 요구사항을 도출하고 요구사항을 기반으로 설계된 SWATT 기반 상호 검증 프로토콜의 특징에 대해 설명한다.

3.1 가정

본 논문이 제안하는 센서 노드 간 상호 검증 프로토콜은 다음과 같은 센서 네트워크 환경을 가정한다.

- 모든 센서 노드는 메모리 내용을 검증할 수 있는 함수를 포함한 동일한 프로그램을 탑재
- 모든 센서 노드는 동일한 하드웨어(CPU, 메모리 등)로 구성
- 공격자는 획득한 센서 노드의 메모리 영역에 제한 없이 접근하여 악의적인 프로그램을 탑재하고 재배치할 수 있으나 센서의 하드웨어 구성은 변경할 수 없음
- 센서 노드의 메모리 부분 중 프로그램 메모리영역(ROM)에 비해 데이터 메모리영역(RAM)의 크기가 매우 작음
- 메모리 영역 검증을 위해 두 센서 노드 간에 교환되는 메시지는 보안을 고려하지 않음
- 센서 노드의 인증 및 키 교환 그리고 이에 수반되는 프로토콜 보안은 본 논문에서 제안하는 검증 프로토콜 성립 후에 이루어짐

공격자는 센서 노드의 메모리 내용에 대한 변형 공격을 통해 센서 노드 자체 또는 센서 네트워크에 대한 공격을 시도한다. 이는 센서 노드의 메모리 크기를 변경하거나 CPU를 교체하는 등 센서 노드의 하드웨어적인 구성 변경을 통한 공격에 비해 메모리 내용의 변경에 의한 공격이 용이하기 때문이다. 공격자는 ROM 영역에 악성 코드를 삽입하고 원래 코드를 RAM 영역으로 이주시켜 악성 코드를 실행시키고 RAM 영역의 코드를 통해 검증 프로토콜을 수행할 수 있다. 즉, RAM 영역의 읽기 속도가 ROM 영역의 읽기 속도보다 빠르기 때문에 공격자는 기존 검증함수를 포함한 프로그램 데이터의 일부 영역을 데이터 메모리 영역에 상주시켜 검증 프로토콜을 수행할 수 있다. 본 논문에서는 센서 노드의 메모리 중 ROM 영역에 비해 RAM 영역의 크기가 작도록 하여, 빠른 RAM 영역 읽기에 의해 공격자가 확보할 수 있는 검증 타이밍에 비해 체크섬 함수 수행을 위해 서로 다른 메모리 영역의 참조를 검사하는 시간이 더 길도록 한다. 따라서 공격자의 빈 RAM 영역을 통한 센서 노드의 공격을 방지할 수 있다.

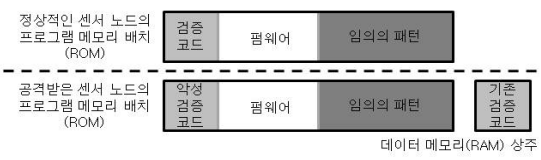
본 논문에서 제안하는 메모리 영역 검증 기법은 센서 노드의 소프트웨어적 무결성을 보장하기 위한 기술이다. 이를 위해 두 센서 노드 간에 교환되는 메시지와 계산되는 메모리 체크섬 값은 보안되지 않은 채널을 통하여 전송된다. 앞선 가정에서 전재했듯이 공격자는 센서 노드의 메모리 영역에 자유롭게 접근할 수 있으며 검증 메시지를 도청할 수 있지만, 센서 노드를 물리적으로 네트워크상에서 제거했을 때와 같은 영향 이상을 줄 수 없다. 따라서 본 논문에서 제안하는 검증 프로토콜의 보안은 요구되지 않는다. 이와 반대로 정상 센서 노드를 가정한 공격은 제안 검증 프로토콜에 의해 검출 가능함을 다음 절에서 보일 것이다. 또한, 무결성이 보장된 센서 노드의 인증과 중요 데이터 전송을 위한 키 교환 등의 절차는 본 논문의 검증 프로토콜 과정 후에 이루어지므로, 본 논문에서는 다루지 않는다.

3.2 위협 모델 및 요구사항

배치된 센서 노드를 획득한 공격자는 센서의 프로그램 메모리 영역에 대한 임의의 읽기와 쓰기가 가능하다. 즉, 공격자가 새롭게 작성한 악의적인 프로그램을 센서의 메모리에 주입하고 재배치하여 센서 또는 센서 네트워크를 공격할 수 있다. 그러나 센서의 하드웨어적인 구성은 변경할 수 없다. 공격자에 의해 변경된 악의적인 센서 노드는 센서 네트워크에 잘못된 데이터를 전송할 수 있거나 전송 데이터의 엿듣기 공격 등에 활용될 수 있다. 이러한 공격을 방지하기 위해서 악의적인 코드가 실행중인 메모리 영역을 가진 센서 노드를 식별하여 네트워크에서 배제할 수 있어야 한다. 다음 (그림 4)는 정상적인 센서 노드의 프로그램 메모리 영역(ROM)의 배치와 공격자에 의해 변경된 센서 노드의 프로그램 메모리 영역(ROM) 및 데이터 메모리 영역(RAM)의 구성을 보인다[6]. 공격자는 악성 검증코드, 즉 바이러스 등의 악성 프로그램을 센서에 주입하고 검증에 사용될 검증코드를 메모리의 데이터 영역에 저장하여 센서 노드의 검증 프로토콜에 사용함으로써 정상적인 검증 과정을 수행할 수 있다. 이러한 방법으로 공격자는 정상적인 검증과정을 거쳐 센서 네트워크에 참여함으로써 지속적으로 센서와 센서 네트워크에 대한 공격을 가할 수 있게 된다.

이러한 공격을 방지하기 위하여 앞선 2장에서 하드웨어/소프트웨어 기반의 디바이스 검증 기법을 설명하였다. 또한, 추가적인 비용이 소요되는 하드웨어적 기법보다는 소프트웨어적인 접근 방식이 센서 네트워크 환경에 적합함을 설명하였다. 그러나 소프트웨어적 기법에 기반 하여 센서 네트워크 환경에 적합한 센서 노드 간 상호 검증 프로토콜을 설계하기 위해서는 다음과 같은 센서 네트워크 환경의 특성이 고려되어야 한다.

- 센서 네트워크 환경에서는 신뢰할 수 있는 검증자가 항상 존재하지 않음
- 1홉(hop)으로 모든 센서 노드를 검증할 수 없음
- 센서 네트워크의 망 구성은 동적으로 변화될 수 있음(센서의 추가, 제거, 이동 등)



(그림 4) 정상적인 센서 노드와 악성 검증코드가 삽입된 센서 노드의 메모리 배치

제안된 검증 프로토콜의 적용을 위한 가정 사항 및 가능한 센서 네트워크 환경에서의 위협 모델, 그리고 고려되어야 할 센서 네트워크 환경의 특징을 반영한 상호 검증 프로토콜 설계를 위한 주요 요구 사항은 다음과 같다.

- 재연 및 선 계산 공격의 저항성: 공격자는 이전에 성공한 attestation의 체크섬을 단순 반복하여 사용할 수 없어야 하며 프로그램 영역의 체크섬을 미리 계산하여 attestation에 사용할 수 없어야 함
- 높은 감지율: 프로그램 메모리 영역의 작은 부분이라도 공격자에 의해 수정될 경우 attestation에 실패해야 함
- 상호 검증: 신뢰할 수 있는 검증자가 없는 센서 네트워크 환경에서 이웃한 노드 간의 상호 검증을 통해 공격받은 센서 노드를 식별할 수 있어야 함
- 검증의 갱신: 상호 검증에 성공한 센서 노드는 검증된 이웃 노드 리스트를 유지하고 주기적으로 검증 결과를 갱신하여 공격자에 의한 센서 네트워크의 공격을 방지

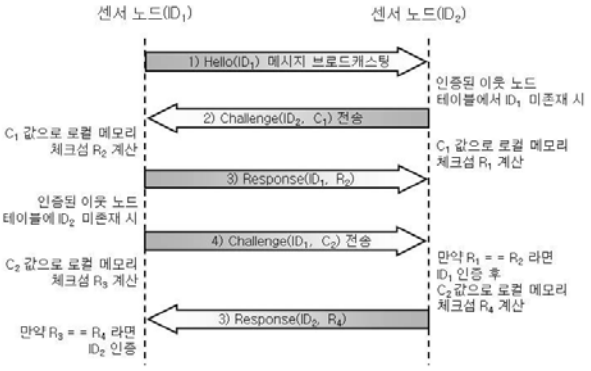
3.3 SWATT 기반 상호 검증 프로토콜 구성

본 장에서는 SWATT 기법에 기반 한 센서 노드 간 상호 검증 프로토콜의 검증 프로세스를 설명한다. 또한, 제안 프로토콜을 통해 검증된 센서 노드로 구성된 센서 네트워크의 상태를 유지하는 과정에 대해 기술한다.

3.3.1 상호 검증 프로토콜 수행 과정

상호 검증 절차는 각 센서 노드가 hello 메시지를 통해 이웃 센서 노드에 자신의 존재를 알림으로써 시작된다. Hello 메시지에는 메시지를 전송하는 센서 노드의 *ID* 정보가 포함된다. Hello 메시지를 전송한 센서 노드의 *ID*가 검증된 이웃 센서 노드 리스트에 존재하지 않는다면 새롭게 네트워크에 진입한 센서로 인식하여 임의의 challenge값과 자신의 *ID*를 hello 메시지를 전송한 센서 노드에게 전송한다. 임의의 challenge값을 사용함으로써 미리 메모리의 체크섬 값을 계산하거나 이전에 계산된 체크섬의 재사용에 의한 공격을 방지할 수 있다. 이 challenge값은 임의의 메모리 위치 생성을 위한 seed로 사용된다.

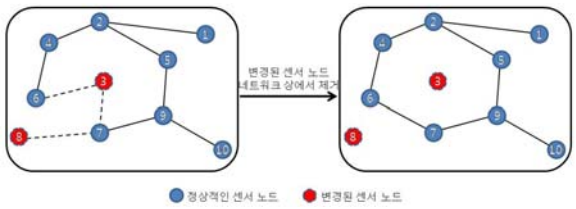
다음 (그림 5)는 본 논문이 제안하는 SWATT를 이용한 센서 노드 간 상호 검증 프로토콜의 수행 과정을 보인다. Seed를 받은 ID_1 노드는 SWATT의 검증 함수인 체크섬 함수를 통해 메모리의 체크섬을 계산하게 된다. Coupon Collector의 문제에 따르면 메모리의 임의적인 접근이라도 모든 메모리를 최소한 한 번은 접근할 수 있는 횟수인 $O(n \ln n)$만큼 메모리 접근을 수행해야 하며[8] 다음 번 메모리 접근 주소의 생성 시 현재의 체크섬 값을 이용하게 하여 병렬계산을 하지 못하도록 한다. 여기서 n은 메모리의 크기이다. 따라서 모든 메모리의 접근을 통해 계산된 체크섬 값은 하나의 메모리 값이 변경되었더라도 예상 값과 차이가 발생되게 된다. 공격자가 기존 검증함수를 데이터 메모리에 상주시켰을 경우를 대비하여 challenge와 response사이에 임



(그림 5) SWATT를 이용한 노드 간 상호 검증 절차

계시간을 정의해야 한다. 일정한 시간 안에 response가 수신되지 않거나 정확한 값이 수신되어도 이 임계시간을 지난 경우에는 attestation이 실패한 것으로 간주한다[6]. 이를 위해 기존 검증함수의 코드 실행시간을 최적화하여 메모리 접근에 대해 검사하는 코드를 추가한 악의적인 검증함수가 더 이상 최적화될 수 없도록 해야 한다. 이 시간 값은 하드웨어 사양에 따라 다르지만 같은 하드웨어로 구성된 센서들은 동일한 시간이 소요되므로 challenge를 전송한 후 자신의 메모리 체크섬 계산을 완료한 시점에서 전송 지연 시간을 더한 시간 안에 상대 센서 노드로부터 response를 수신해야 한다. 여기서 전송 지연 시간은 악의적인 검증함수로 인해 증가된 시간보다는 작아야 한다. ID_1 노드가 체크섬 계산을 수행하는 동안, ID_2 노드도 자신의 메모리 체크섬 계산을 수행한다. 이때 체크섬 계산의 수행 횟수는 모든 센서 노드가 동일한 메모리 크기를 가지고 있기 때문에 ID_1 센서 노드와 동일하다. 만약 ID_1 센서 노드가 정상적인 센서 노드라면 ID_2 센서 노드가 계산한 체크섬 값과 동일한 값을 response하게 되어 ID_2의 검증된 센서 노드 리스트에 저장되게 된다. 이후 센서 노드 ID_2는 ID_1 센서 노드로부터의 다음 번 hello 메시지를 기다린다. ID_1 센서 노드는 response를 전송한 후 검증된 이웃 센서 노드 리스트를 검색하여 ID_2가 존재하지 않으면 앞서 설명된 절차와 동일하게 임의의 challenge를 ID_2로 전송하여 ID_2와 상호 검증을 수행한다. 이와 같이 hello 메시지를 전송한 노드가 response을 전송한 후 바로 상대편 노드와 검증 과정을 수행하게 하여 수행하지 않을 때보다 상호 통신이 가능해질 때까지의 시간을 줄일 수 있다.

센서 노드는 이웃한 센서 노드들과의 통신을 통해 네트워크를 구성하며 추가, 제거 및 이동이 가능하기 때문에 센서 네트워크는 동적으로 구성된다. 또한 각 센서 노드들은 이웃하지 않은 노드들과는 애드훅(ad hoc) 기능을 통해 통신을 한다. 앞서 설명했듯이 센서 네트워크 망 구성 시 주기적인 메시지 브로드캐스팅을 통하여 자신의 존재를 이웃한 센서 노드들에게 알려 상호 검증을 수행한 후 센서 네트워크에 참여한다.



(그림 6) 제안 프로토콜을 적용한 센서 네트워크

(그림 6)은 제안 프로토콜을 통해 구성될 수 있는 센서 네트워크의 일부를 보인다.

센서 노드 3과 8은 공격자에 의한 메모리 변경 공격을 통해 메모리가 변경되었으며, 공격 받은 시점 직후의 검증 프로토콜을 올바르게 수행할 수 없으므로 센서 네트워크에 참여하지 못하는 모습을 보여준다.

3.3.2 검증된 센서 네트워크 유지

각 센서 노드들은 주기적으로 hello 메시지를 이웃 센서 노드에 브로드캐스팅한다. Hello 메시지를 브로드캐스팅하는 주기는 공격자가 센서 노드를 획득하여 악의적인 코드를 주입하고 네트워크에 추가시키는데 소요되는 최소의 시간($T_{\min}$)보다는 작아야 한다[9][10]. 그래야만 공격자에 의해 변형된 센서 노드를 이웃 노드들이 검증 프로토콜 과정에서 식별할 수 있게 된다. 즉, 다음 hello 메시지의 수신 예상시간 안에 해당 hello 메시지를 수신하지 못한다면 해당 센서 노드는 공격자에 의해 제거되었거나 다른 곳으로 이동한 것으로 인식하여 검증된 이웃 센서 노드 리스트에서 제거한다. 따라서 만약 공격자가 센서 노드의 프로그램 메모리를 변경하여 $T_{\min}$ 시간 후에 네트워크에 추가한다면 이웃한 센서 노드들과 상호 검증 절차를 수행하게 되어 검증에 실패하게 된다. 상호 검증에 실패한 센서 노드는 네트워크에 참여할 수 없게 되므로 검증된 센서 노드로 구성된 센서 네트워크를 유지할 수 있다. Jing Deng은 실험을 통해 공격자가 노드를 획득한 후 메모리의 데이터를 얻기까지의 시간이 수십 초안에 이루어짐을 제시하였다[9]. 따라서 $T_{\min}$은 수십 초안에 이루어질 수 있으므로 hello 메시지의 브로드캐스팅 주기는 이 시간보다는 작아야 한다.

모든 센서 노드들은 이웃한 센서 노드 중 공격자에 의해 변경되지 않은 정상적인 센서 노드들, 즉 검증된 이웃 센서 노드 리스트를 유지한다. 이 리스트의 센서 노드 목록은 해당 센서 노드와의 상호 검증이 성공했음을 보장하며 정해진 시간 안에 hello 메시지가 수신되지 않거나 상호 검증이 실패한 경우에 리스트에서 삭제된다. 또한 리스트의 각 센서 노드 당 하나의 타이머를 두어, 검증이 성공한 이후 타이머가 생성되고 카운트가 시작되며 최대 카운트 값은 $T_{\min}$으로 고정된다. 최대 카운트 값인 $T_{\min}$ 안에 해당 센서 노드로부터 hello 메시지가 수신되면 타이머는 초기화되고 다음 hello 메시지를 기다리게 된다. 이웃한 센서 노드로부터 hello 메시지를 수신한 센서 노드는 검증된 이웃 센서 노드 리스트를 검사한다. 해당 센서 노드가 리스트에 존재한다면 이미 상호 검증이 성공한 센서 노드이므로 검증 절차를 수행할 필요가 없고 존재하지 않는다면 두 센서 노드 간 상호 검증 절차를 수행하게 된다. 따라서 SWATT나 기본 임계값 비밀 공유와 과반수 투표에 기반 한 검증 방식에서는 보장할 수 없는 검증 수행 후에 센서 노드가 손상되지 않았음을 검증결과의 유지를 통해 보장할 수 있다. 즉, 다른 3가지 방식에서는 검증 결과를 유지하지 않아 검증 수행 후에 공격받지 않았음을 보장하지 못해 필요 시 다시 검증 절차를 수행하여 정상적인 노드임을 확인해야만 한다.

4. 보안검증 및 성능분석

제안된 센서 노드 간 상호 검증 프로토콜은 3장에서 설명된 요구사항을 만족하도록 설계되어야 한다. 이번 장에서는 제안 검증 프로토콜의 보안성을 검증하고 검증 수행 성능에 대해 분석한다.

4.1 보안검증

제안된 프로토콜은 센서 간 상호 검증을 통해 정상적인 센서만으로 네트워크를 구성할 수 있다. 이를 위해서는 악의적인 공격으로 인해 변경된 메모리를 가진 센서와의 상호 검증이 실패해야 하며, 이미 언급된 제안 검증 프로토콜의 네트워크 유지 기능을 통하여 공격으로 인한 변경이 예상되는 센서 노드를 구별할 수 있어야 한다.

- **재연 및 선 계산 공격의 저항성:** Challenge를 seed로 한 임의의 메모리 번지 순회 기법을 적용하여 변경된 메모리의 위치가 언제 접근되지는 알 수 없게 하기 때문에 미리 계산된 체크섬을 통한 공격과 이전 체크섬을 통한 공격에 강하다. 공격자는 변경한 부분의 원래 메모리 값들을 비어있는 데이터 메모리 영역에 적재하고 검증함수를 수정하여 변경된 메모리 영역에 대한 접근을 옮겨진 메모리 영역으로의 접근으로 전환하여 예상 체크섬과 동일한 값을 반환할 수 있다. 이를 해결하기 위해 제안 검증 프로토콜은 모든 메모리 접근 시 메모리 번지 비교를 수행해야 하는 악의적인 검증함수의 실행시간의 overhead 특성을 사용한다. 즉, 예상시간 이후에 도착한 response는 예상 값과 동일하더라도 attestation에 실패한 것으로 판단하는 SWATT 기법의 장점을 센서 네트워크 환경에 적용한다. 하지만 정상적인 센서 노드일지라도 네트워크 전송지연 시간으로 인해 response가 예상시간보다 늦게 도착하여 검증에 실패하는 경우가 발생하게 된다. 그러나 주기적인 hello 메시지로 인해 결국엔 검증에 성공하게 되고 한번 검증에 성공하여 검증 리스트에 등록된 노드는 토폴로지 변화가 발생하기 전까지는 주기적으로 리스트를 갱신만 하면 된다.
- **높은 감지율:** SWATT 기법은 challenge를 seed로 사용해 임의의 메모리 번지를 순회하며 체크섬을 계산하여

모든 메모리를 한 번 이상 접근할 수 있도록 $O(n \ln n)$번 체크섬을 반복적으로 수행한다. 따라서 임의의 한 영역의 메모리 값이 변경될 경우 최종 체크섬 값은 원래 메모리의 체크섬과 다르므로 작은 메모리 변경에도 높은 감지율을 유지할 수 있다.

- **상호 검증:** 센서의 초기 배치나 미검증된 센서 노드가 네트워크에 참여하고자 한다면 자신과 이웃한 센서 노드들과 검증 프로토콜을 수행하여 정상적인 센서 노드로 검증받아야 한다. 하지만 이웃한 센서 노드들 역시 공격자에 의해 손상된 센서 노드일 확률은 동일하기 때문에 이웃한 센서 노드들 간 상호 검증을 수행함으로써 악의적인 공격에 의해 변경된 센서 노드와의 네트워크 구성을 차단할 수 있다.
- **검증의 갱신:** 정상적인 센서 노드로 검증되었더라도 언제든 공격자가 해당 센서 노드의 프로그램 메모리를 변경하여 재배치할 수 있다. 공격자가 센서를 획득하여 악의적인 프로그램을 적재하고 네트워크에 재배치하기까지는 $T_{\min}$이라는 시간이 필요하기 때문에 이 시간보다 작은 주기로 검증 리스트를 갱신하게 함으로써 갱신 주기를 초과하여 hello 메시지를 전송한 노드와 상호 검증을 수행하도록 한다. 즉, 주기적인 hello 메시지를 시작으로 상호 검증 프로토콜을 수행함으로써 공격자에 의해 재배치된 센서 노드와의 상호 검증과정을 통해 해당 노드를 네트워크에서 배제할 수 있다.

제안된 프로토콜은 신뢰받은 검증자가 검증을 수행해야 하는 SWATT에 단점을 상호 검증을 통해 극복하였고 신뢰받은 검증자가 검증할 모든 센서를 1hop으로 검증하기 어려운 문제도 이웃한 노드들 간 1hop으로 모든 센서들을 검증할 수 있게 하였다. 또한 기본 임계값 비밀 공유 방식은 검증을 수행할 클러스터 헤더가 공격받은 경우 검증 결과를 신뢰할 수 없으나 제안된 프로토콜은 상호 검증을 통해 해결하였으며 과반수 투표에 기반 한 검증 방식의 문제인 이웃한 노드들이 과반수 이상 공격받아 잘못된 검증 결과를 유도하는 문제도 일대일 방식으로 해결하였다.

4.2 성능분석

이웃 센서 노드로부터의 challenge에 대해 체크섬을 계산하기 위해 메모리에 접근해야 하는 횟수는 메모리의 크기를 ms라하면 $O(ms \ln ms)$이 된다. 한 센서를 검증하기 위해서는 challenge를 받은 센서뿐만이 아니라 challenge를 전송한 센서도 체크섬 계산을 수행한다. 따라서 새롭게 네트워크에 진입한 센서가 정상적인 센서 노드라면 상호 검증 프로토콜을 수행해야 하기 때문에 센서 간 4번의 체크섬 계산이 필요하고 이웃노드의 수를 n이라하면 총 $4n(ms \ln ms)$ 번의 메모리 접근이 요구된다. 만약 공격으로 인해 손상된 센서 노드가 네트워크에 참여하였다면 상호 검증 프로토콜은 수행되지 않으므로 $2n(ms \ln ms)$번의 메모리를 접근해야 한다.

또한 제안된 프로토콜과 SWATT, 기본 임계값 비밀 공유 방식과 과반수 투표에 기반한 검증 방식의 총 메모리 접근 횟수는 <표 2>와 같다.

즉, SWATT는 한 센서 노드를 검증하기 위해 2번의 체크섬이 필요하고 $n+1$개의 센서 노드들을 검증하기 위해서는 총 $2(n+1)(ms \ln ms)$의 메모리 접근이 요구된다. 기본 임계값 비밀 공유 방식은 $n+1$번의 다항식 계산과 많은 계산이 필요한 1번의 내삽법이 필요하고 또한 2번의 해시 계산과 블록기반 체크섬이 필요하다. 그러므로 $n+1$개의 센서 노드들을 검증하기 위해서는 총 $2(n+1)(ms \ln ms)$의 메모리 접근이 요구된다. 여기서 블록기반 체크섬일지라도 메모리 접근 횟수는 비블록기반 체크섬과 동일하다. 그리고 과반수 투표에 기반 한 검증 방식은 n개의 이웃한 노드들과 검증을 수행해야하므로 $n+1$개의 센서 노드들을 검증하기 위해서는 총 $(n+1)2n(ms \ln ms)$의 메모리 접근이 요구된다. 그러나 제안된 상호 검증 프로토콜은 1번의 상호 검증을 통해 이웃한 노드들과 검증을 수행하므로 총 $4n(ms \ln ms)$의 메모리 접근으로 n+1개의 센서 노드들을 검증할 수 있다. 따라서 제안된 프로토콜이 과반수 투표에 기반 한 검증 방식보다는 메모리 접근 횟수가 적지만 다른 두 방식보다는 많음을 알 수 있다.

정상적인 센서가 새롭게 네트워크에 진입한 경우 그 센서의 이웃 센서 노드의 수를 n이라하면 1번의 hello메시지 브로드캐스팅과 센서 간 2번의 challenge-response 메시지가 송수신되어 총 $1+4n$번의 메시지 전송이 필요하게 된다. 또한 이미 검증된 이웃 센서 노드와는 주기적인 hello메시지의 전송만이 필요하다. 만약 새롭게 네트워크에 진입한 센서 노드가 공격자에 의해 손상된 센서 노드라면 진입한 센서 노드와의 검증이 실패하게 되어 상대편 센서 노드의 검증 프로토콜은 더 이상 수행되지 않으므로 총 $1+2n$번의 메시지가 송수신되어야 한다. 센서 노드 간 상호 검증이 수

〈표 1〉 센서 노드 간 상호 검증을 위한 메모리 접근 횟수

구분	메모리 접근 횟수
정상적 센서 노드와의 검증 수행	$4n(ms \ln ms)$
변경된 센서 노드와의 검증 수행	$2n(ms \ln ms)$

〈표 2〉 센서 노드의 검증을 위한 총 메모리 접근 횟수 비교

구분	총 메모리 접근 횟수
제안된 상호 검증 프로토콜	$4n(ms \ln ms)$
SWATT	$2(n+1)(ms \ln ms)$
기본 임계값 비밀 공유 방식	$2(n+1)(ms \ln ms)$
과반수 투표에 기반 한 검증 방식	$(n+1)2n(ms \ln ms)$

〈표 3〉 센서 노드 간 상호 검증을 위한 통신 오버헤드

구분	메시지 전송 횟수
정상적 센서 노드와의 검증 수행	$1+4n$
변경된 센서 노드와의 검증 수행	$1+2n$

행된 이후에는 센서 노드가 위치를 변경하지 않는 이상 주기적인 hello 메시지 브로드캐스팅을 통해 검증된 이웃 노드 리스트를 참조하기 때문에 통신 오버헤드를 줄일 수 있다. 또한 hello 메시지 전송으로 인한 오버헤드는 $T_{\min}$에 의존적으로 수십 초의 주기를 가지고 있고 적은 양의 byte로 구성되어 있어 네트워크의 전체 메시지에서 매우 작은 부분을 차지한다.

또한 제안된 프로토콜과 SWATT, 기본 임계값 비밀 공유 방식과 과반수 투표에 기반 한 검증 방식의 통신 오버헤드는 <표 4>와 같다. 즉, SWATT는 검증자와 센서 노드 간 1번의 challenge-response 메시지가 필요하여 2번의 메시지 전송이 요구되고 $n+1$개의 센서 노드들을 검증하기 위해서는 $2(n+1)$개의 메시지가 필요하다. 기본 임계값 비밀 공유 방식은 1개의 노드를 검증하기 위해 1번의 hello메시지 브로드캐스팅과 비밀 공유를 위해 n번의 비밀 공유의 분배, seed 복구를 위해 최소한 $(k-1)$개의 메시지와 1번의 challenge-response메시지가 송수신되어 $3+n+(k-1)$번의 메시지 전송이 요구되고 $n+1$개의 센서 노드들을 검증하기 위해서는 $(n+1)(3+n+(k-1))$개의 메시지가 필요하다. 그리고 과반수 투표에 기반한 검증 방식은 1개의 센서 노드를 검증하기 위해 1번의 hello메시지 브로드캐스팅과 센서 간 n번의 정보 분배와 n번의 challenge-response메시지가 송수신되어 $1+3n$번의 메시지 전송이 요구되고 $n+1$개의 센서 노드들을 검증하기 위해서는 $(n+1)(1+3n)$개의 메시지가 필요하다. 그러나 제안된 상호 검증 프로토콜은 1번의 상호 검증을 통해 이웃한 노드를 포함한 식별된 모든 노드들의 검증을 수행하므로 총 $1+4n$번의 메시지만이 필요하다. 따라서 신뢰받은 검증자를 통해 모든 노드들을 검증하는 SWATT를 제외하고는 제안된 프로토콜이 통신 오버헤드가 적음을 알 수 있다.

제안된 상호 검증 프로토콜은 이미 검증된 이웃 센서 노드 리스트를 메모리에 유지해야 한다. 따라서 센서 노드 *ID* 저장에 필요한 메모리 크기를 *SID*라하고 S/W 타이머 설정을 위한 메모리 크기를 *ST*라 한다면 검증된 이웃 센서 노드의 수만큼의 *SID* 영역과 *ST* 영역이 필요하다. 즉 검증된 이웃 센서 노드의 수가 n일 때의 메모리 요구는 아래 식 (2)와 같이 계산될 수 있다.

〈표 4〉 이웃한 노드를 포함한 센서 노드들의 검증을 위한 통신 오버헤드 비교

구분	메시지 전송 횟수
제안된 상호 검증 프로토콜	$1+4n$
SWATT	$2(n+1)$
기본 임계값 비밀 공유 방식	$(n+1)(3+n+(k-1))$ ※ k : 다항식의 차수
과반수 투표에 기반한 검증 방식	$(n+1)(1+3n)$

$$\text{요구되는 메모리} = n(SID+ST) \quad (2)$$

5. 결 론

제안된 센서 노드 간 상호 검증 프로토콜은 challenge-response 방식을 사용하는 SWATT 기법을 기반으로 설계되었다. SWATT는 원격에서 소프트웨어를 통한 효과적인 임베디드 장비 검증 기법으로 추가적인 하드웨어 비용이 필요하지 않다. 본 논문에서는 SWATT 기법을 이용하여 센서 네트워크의 특징적 요구사항을 반영한 센서 노드 간 상호 메모리 검증 프로토콜을 제안하였다. 또한, 제안된 상호 검증 프로토콜의 보안성 검증과 성능 분석을 수행하고 제안된 검증 프로토콜을 통하여 검증된 센서 노드만으로 센서 네트워크가 구성되며 공격으로 인해 메모리 영역이 변경된 센서 노드는 네트워크에서 배제됨을 확인하였다. 이미 검증된 센서 노드도 임의의 시간에 공격자에 의한 공격이 가능하기 때문에 검증된 이웃 센서 노드 리스트를 유지하여 주기적인 hello 메시지만으로 검증 결과를 갱신하여 검증 프로토콜의 성능을 확보하였다. 또한, 주기적 상호 검증 프로토콜의 수행을 통해 공격자에 의해 변경된 뒤 재배치된 센서 노드로부터 네트워크의 보안성을 확보하였다.

참 고 문 헌

[1] E. Shi, and A. Perrig, “Designing secure sensor networks”, IEEE Wireless Communication, pp.8 - 43, 2004.

[2] C. Krauß, F. Stumpf, and C. Eckert, “Detecting Node Compromise in Hybrid Wireless Sensor Networks Using Attestation Techniques”, ESAS 2007, LNCS 4572, pp.203 - 217, 2007.

[3] Group, T.C.: Trusted Platform Module (TPM) specifications, Technical report (2006) https://www.trustedcomputinggroup.org/specs/TPM

[4] Microsoft, NGSCB(Next-Generation Secure Computing Base), http://www.microsoft.com/resources/ngscb/default.mspx

[5] Y. Yang, X. Wang, S. Zhu, and G. Cao, “Distributed Software-based Attestation for Node Compromise Detection in Sensor Networks”, SRDS 2007, pp.219-230, 2007.

[6] A. Seshadri, A. Perrig, L. V. Doorn, and P. K. Khosla,

"SWATT: SoftWare-based ATTestation for Embedded Devices", IEEE Symposium on Security and Privacy, pp.272 - 282, 2004.

[7] A. Seshadri, M. Luk, E. Shi, A. Perrig, L. V. Doorn, and P. K. Pioneer, "Verifying integrity and guaranteeing execution of code on legacy platforms", SOSP, pp.1 - 15, 2005.

[8] M. Mitzenmacher, and E. Upfal, "Probability and Computing: Randomized Algorithms and Probabilistic Analysis", Cambridge University Press, 2005.

[9] S. Zhu, S. Setia, and S. Jajodia, "LEAP: efficient security mechanisms for large-scale distributed sensor networks", CCS '03, pp 62 - 72, 2003.

[10] J. Deng, R. Han, and S. Mishra, "A practical study of transitory master key establishment for wireless sensor networks", SecureComm 2005, pp.289 - 299, 2005.

허 경 수

e-mail : ksheo@lignex1.com
2004년 서강대학교 컴퓨터학과(학사)
2004년~현재 LIG넥스원 재직
2007년~현재 성균관대학교 이동통신 공학과 석사과정
관심분야 : 임베디드 시스템, 운영체제, 네트워크 보안

최 현 우

e-mail : iskraskk@ece.skku.ac.kr
2006년 성균관대학교 전기전자컴퓨터공학부(학사)
2006년~현재 성균관대학교 전기전자 컴퓨터공학부(석사과정)
관심분야 : 모바일 에이전트, 네트워크 보안, 유비쿼터스 컴퓨팅

장 현 수

e-mail : jhs4071@ece.skku.ac.kr
2002년 성균관대학교 전기전자및컴퓨터공학과(학사)
2005년 성균관대학교 전기전자 및 컴퓨터공학과(공학석사)
2007년~현재 성균관대학교 전기전자 및 컴퓨터공학과(박사과정)
관심분야 : 유비쿼터스 컴퓨팅, 이동 에이전트, HCI, 미들웨어 등

엄 영 익

e-mail : yieom@ece.skku.ac.kr
1983년 서울대학교 계산통계학과(학사)
1985년 서울대학교 전산과학과 (이학석사)
1991년 서울대학교 전산과학과 (이학박사)
2000년~2001년 Dept. of Info. and Comm. Science at UCI 방문교수
2004년~2005년 한국정보보호학회 논문지 편집이사
2005년~2006년 한국정보처리학회 학회지 편집위원장(상임이사)
1993년~현재 성균관대학교 정보통신공학부 교수
2007년~현재 성균관대학교 정보통신처 처장
관심분야 : 분산 컴퓨팅, 이동 컴퓨팅, 이동 에이전트, 시스템 보안, 운영체제, 내장형 시스템 등