공학박사학위논문

무선 센서 네트워크에서 브로드캐스트 인증을 위한 효율적인 일회용 서명 기법

An Efficient One-time Signature Scheme for Broadcast Authentication in Wireless Sensor Networks

2013년 2월

서울대학교 대학원
전기 · 컴퓨터 공학부
이 재 흥

공학박사학위논문

무선 센서 네트워크에서 브로드캐스트 인증을 위한 효율적인 일회용 서명 기법

An Efficient One-time Signature Scheme for Broadcast Authentication in Wireless Sensor Networks

2013년 2월

서울대학교 대학원
전기 · 컴퓨터 공학부
이 재 흥

무선 센서 네트워크에서 브로드캐스트 인증을 위한 효율적인 일회용 서명 기법

An Efficient One-time Signature Scheme for Broadcast Authentication in Wireless Sensor Networks

지도교수 조유근

이 논문을 공학박사 학위논문으로 제출함
2012년 11월

서울대학교 대학원
전기 · 컴퓨터 공학부
이 재 흥

이재흥의 공학박사 학위논문을 인준함
2012년 12월

위원장	신 현 식	(인)
부위원장	조 유 근	(인)
위 원	박 근 수	(인)
위 원	민 상 렬	(인)
위 원	박 용 수	(인)

초 록

무선 센서 네트워크에서 브로드캐스트 인증은 가장 기본적이면서도 중요한 보안 요소이다. 하지만 센서 노드의 자원 제약적인 특성으로 인해 기존의 인증 기법들은 대부분의 경우 무선 센서 네트워크에서 브로드캐스트 인증을 위해 사용될 수 없다. 일대일 통신에서 사용했던 MAC을 통한 방법은 같은 키를 가진 수신자가 메시지를 위조할 수 있기 때문에 부적절하고, 공개 키 기반 서명 기법은 너무 많은 계산량을 요구한다. μTESLA는 노드들 간의 시간 동기화를 필요로 할 뿐만 아니라 인증 시간의 지연도 발생한다.

무선 센서 네트워크에서 브로드캐스트 인증은 일회용 서명 기법을 통해서도 가능하다. 일회용 서명 기법은 일방향 함수만을 기반으로 하므로 공개 키 기반 서명 기법들보다 서명 생성과 검증에 많은 계산을 필요로 하지 않아 무선 센서 네트워크 환경에 적합하다. 하지만 주어진 비밀 키로 오직 한 번만 서명이 가능하고, 공개 키의 저장을 위해 많은 양의 저장 공간을 필요로 하며 서명의 크기가 커서 높은 통신 부하를 야기한다.

본 논문은 HORSIC이라 부르는 무선 센서 네트워크에서 브로드캐스트 인증을 위한 효율적인 일회용 서명 기법을 제안한다. HORSIC은 기존에 제안된 HORS의 확장이다. HORS는 t개의 원소를 가지는 비밀 키 집합에서 암호학적 해시 함수 H를 통해 서명하고자 하는 메시지에 따라 서로 다른 k개의 원소를 선택하여 이들로 서명을 구성한다. HORSIC도 HORS와 같은 방법으로 암호학적 해시 함수 H를 통해 k개의 원소를 선택하지만, 그 후 또 다른 암호학적 해시 함수 G와 전단사 함수 $C_{k,z}$를 통해 서명하고자 하는 메시지에 따라 서로 다른 횟수의 일방향 함수를 앞에서 선택한 k개의 원소 각각에 적용하여 이들로 서명을 구성한다.

HORSIC을 제외한 기존 일회용 서명 기법들의 서명 위조 확률은 공개 키 크기와 서명 크기에 의해 결정된다. 따라서 이러한 기법들의 경우 보안성을 낮추지 않으면서 공개 키 크기와 서명 크기를 동시에 줄이는 것은 불가능하다. 하지만 HORSIC은 서명하고자 하는 메시지에 따라 비밀 키에 적용되는 일방향 함수의 횟수를 서로 다르게 함으로써 서명 위조 확률을 줄였으며, 따라서 기존 일회용 서명 기법들에 비해 작은 크기의 공개 키나 서명을 사용하면서도 같은 수준의 보안성을 제공할 수 있다. 대신 키 생성과 서명 검증 비용은 기존 일회용 서명 기법들에 비해 큰 편이다. 하지만 센서 노드에서 전체적으로 소모되는 에너지의 양은 HORSIC의 경우 서명의 크기가 줄어듬으로 인해 다른 일회용 서명 기법들에 비해 오히려 더 적은 경우가 많다.

비록 HORSIC이 무선 센서 네트워크에서 브로드캐스트 인증을 위해 설계된 일회용 서명 기법이기는 하지만, 스마트그리드와 같이 제한된 자원을 가진 노드들로 이루어진 다른 다양한 네트워크에서도 사용 가능하다.

주요어: 무선 센서 네트워크, 암호학, 인증, 브로드캐스트 인증, 일회용 서명

학번: 2007-30237

목 차

표 목 차

그 림 목 차

제 1 장 서론

1.1 연구 배경

무선 센서 네트워크는 특정 지역에 분포된 초소형의 센서 노드들이 온도나 소리, 압력 같은 주변의 물리적 또는 환경적 조건들을 관찰, 수집한 뒤 사용자가 원하는 형태로 가공하여 무선 통신을 통해 전달해 주는 네트워크이다 [ASSC02, AK04]. 무선 센서 네트워크는 보통 수백에서 수천 개의 센서 노드들로 구성되기 때문에 각 센서 노드는 가능한 한 최소한의 비용으로 제작되어야 하며, 따라서 매우 제한적인 컴퓨팅 파워와 메모리를 가진다. 게다가 센서 노드들은 작은 크기의 배터리를 통해 전원을 공급받는 방식을 많이 사용하므로 오래 사용하기 위해서는 가능한 한 에너지 효율적으로 설계되어야 한다.

무선 센서 네트워크는 군사 전투 지역과 같은 적대적인 환경에서도 사용될 수 있으므로 보안이 매우 중요하다. 무선을 통신 수단으로 사용하기 때문에 공격자가 메시지를 쉽게 도청할 수도 있을 뿐만 아니라, 위조된 메시지를 보내는 것도 가능하다. 따라서 메시지의 위조 또는 변조를 방지하기 위한 인증은 무선 센서 네트워크에 있어 가장 기본적이면서도 중요한 요소이다.

일대일 통신의 경우 전송되는 메시지에 MAC이라 부르는 메시지 인증 코드 (Message Authentication Code)를 첨부함으로써 인증을 수행할 수 있다 [MVOV97, KSW04, DHHM05]. 송신자와 수신자는 둘만이 알고 있는 비밀 키를 공유하며 송신

자는 이 키를 사용해 보내고자 하는 메시지마다 그에 해당하는 MAC을 생성한 뒤 메시지에 첨부해 이를 수신자에게 보낸다. 그럼 수신자는 메시지에 첨부된 MAC을 검증함으로써 그 메시지가 송신자로부터 왔고 변조되지 않았음을 확인할 수 있다. 이러한 MAC은 매우 제한적인 자원을 가지는 센서 노드에서도 효율적으로 생성하고 검증하는 것이 가능하다 [KKP99, HSW+00, PST+02].

무선 센서 네트워크에서는 하나의 노드에서 네트워크 상의 모든 노드로 패킷을 전달하는 브로드캐스트 (broadcast) 통신이 일대일 통신보다 더 자주 사용된다. 이러한 통신의 경우에도 인증 기능이 필요한데 이 때 일대일 통신에서 사용했던 MAC을 통한 방법은 적절하지 않다 [BDF01, LPW06]. 브로드캐스트 인증에서는 수신자가 여러 명인데 이들이 송신자와 같은 키를 공유하므로 그 중 누구라도 송신자를 가장하여 메시지를 보낼 수 있기 때문이다. 그러므로 브로드캐스트 인증을 위해서는 오직 송신자만 인증 정보를 생성할 수 있고 수신자들은 인증 정보를 검증만 할 수 있도록 하는 비대칭적 요소가 포함되어야만 한다.

이러한 비대칭적 요소를 제공하기 위한 가장 일반적인 방법은 RSA [RSA78] 나 DSA [LG09] 같은 공개 키 기반 전자 서명 기법들을 사용하는 것이다. 특히 타원곡선 암호 (Elliptic Curve Cryptography) 는 작은 서명 크기와 빠른 계산으로 인해 에너지를 아낄 수 있어서 무선 센서 네트워크에 적합하다 [MWS04, GPW+04, PLP06, SOS+08]. 하지만 DSA의 타원곡선 버전인 ECDSA (Elliptic Curve Digital Signature Algorithm) 도 자원 제한적인 센서 노드들에게는 너무 많은 계산량을 요구한다.

μTESLA [PST+02] 는 오직 대칭적인 요소들만을 사용해서 효율적인 브로드캐스트 인증을 수행한다. 이는 일방향 키 체인과 지연 키 공개 기법을 통해 가능한데, 센서 노드들 간의 시간 동기화를 필요로 하며 인증 시간의 지연이 발생하는 단점을 가진다.

μTESLA에 대해서는 2.1절에서 자세히 살펴볼 것이다.

무선 센서 네트워크에서 브로드캐스트 인증은 일회용 서명 기법들을 통해서도 가능하다 [DH76, Rab78, Lam79, Mer88, Mer89, EGM89, BM94, BM96, Roh99, Per01, MP02, RR02, BGD$^+$06, PC06, CSLH06, LC11]. 일회용 서명 기법들은 일방향 함수만을 기반으로 하므로 RSA [RSA78]나 DSA [LG09] 같은 일반적인 공개 키 기반 서명 기법들보다 서명 생성과 검증에 많은 계산을 필요로 하지 않아 무선 센서 네트워크 환경에 적합하다. 또한 μTESLA와는 달리 송신자와 수신자들간의 시간 동기화를 필요로 하지도 않으며, 수신자가 데이터 패킷을 받자마자 바로 인증을 수행할 수도 있다. 더구나 기존의 일반적인 공개 키 기반 서명 기법들은 양자 컴퓨터 (quantum computer) [NCG02]가 본격적으로 실용화 된다면 더 이상 사용할 수 없게 되지만 [Sho99], 일방향 함수를 기반으로 하는 일회용 서명 기법들은 여전히 사용 가능하다 [Gro96]. 하지만 이름 그대로 주어진 비밀 키로 오직 한 번만 서명이 가능하고, 공개 키의 저장을 위해 많은 양의 메모리를 필요로 하며 서명의 크기가 커서 높은 통신 부하를 야기한다. 그러므로 무선 센서 네트워크에서 브로드캐스트 인증을 위해 일회용 서명 기법을 사용하기 위해서는 이러한 단점들을 보완할 필요가 있다.

1.2 연구 목적 및 범위

본 논문에서는 HORSIC (Hash to Obtain Random Subset and Integer Composition)이라 부르는 무선 센서 네트워크에서 브로드캐스트 인증을 위한 효율적인 일회용 서명 기법을 제안한다. HORSIC은 기본적으로 Reizin이 제안한 HORS [RR02]의 확장이다.

- HORS는 t개의 원소를 가지는 비밀 키 집합에서 암호학적 해시 함수 H를 통해

서명하고자 하는 메시지에 따라 서로 다른 k 개의 원소를 선택하여 이들로 서명을 구성한다.

- HORSIC도 HORS와 마찬가지로 t 개의 원소를 가지는 비밀 키 집합에서 암호학적 해시 함수 H 를 통해 서명하고자 하는 메시지에 따라 서로 다른 k 개의 원소를 선택한다. 그리고 나서 또 다른 암호학적 해시 함수 G 와 전단사 함수 $C_{k,z}$ 를 통해 서명하고자 하는 메시지에 따라 서로 다른 횟수의 일방향 함수를 앞에서 선택한 k 개의 원소 각각에 적용하여 이들로 서명을 구성한다.

위에서 언급한 특징으로 인해 HORSIC은 HORS나 그 밖의 다른 일회용 서명 기법들과는 달리 공개 키 크기와 서명 크기를 동시에 줄이는 것이 가능하다. 대신 키 생성과 서명 검증에 걸리는 시간이 증가하는 단점이 있다. HORSIC은 서명을 하고자 하는 메시지에 따라 비밀 키에 적용되는 일방향 함수의 횟수를 서로 다르게 하는데 이는 Merkle의 논문 [Mer88, Mer89]에 나오는 Winternitz의 아이디어이다. Winternitz의 아이디어에 관해서는 2.2.2절에서 자세히 살펴볼 것이다.

비록 HORSIC이 무선 센서 네트워크에서 브로드캐스트 인증을 위해 설계된 일회용 서명 기법이기는 하지만, 스마트그리드 [LC11]와 같이 제한된 자원을 가진 노드들로 이루어진 다른 다양한 네트워크에서도 사용 가능하다.

그림 1.1은 본 논문에서 제안하는 일회용 서명 기법이 가정하고 있는 센서 네트워크 모델을 나타낸다. 센서 네트워크는 하나의 베이스 스테이션과 많은 수의 센서 노드들로 구성되며, 센서 노드들은 베이스 스테이션에 비해 매우 제한된 자원만을 가진다. 또한 [CSLH06]에서와 마찬가지로 문제를 단순화하기 위해 베이스 스테이션만이 메시지를 브로드캐스트할 수 있다고 가정하였다. 만약 센서 노드가 메시지를 브로드캐스트하고자 한다면 센서 노드는 일대일 통신을 통해 메시지를 베이스 스테이션에게 보내고

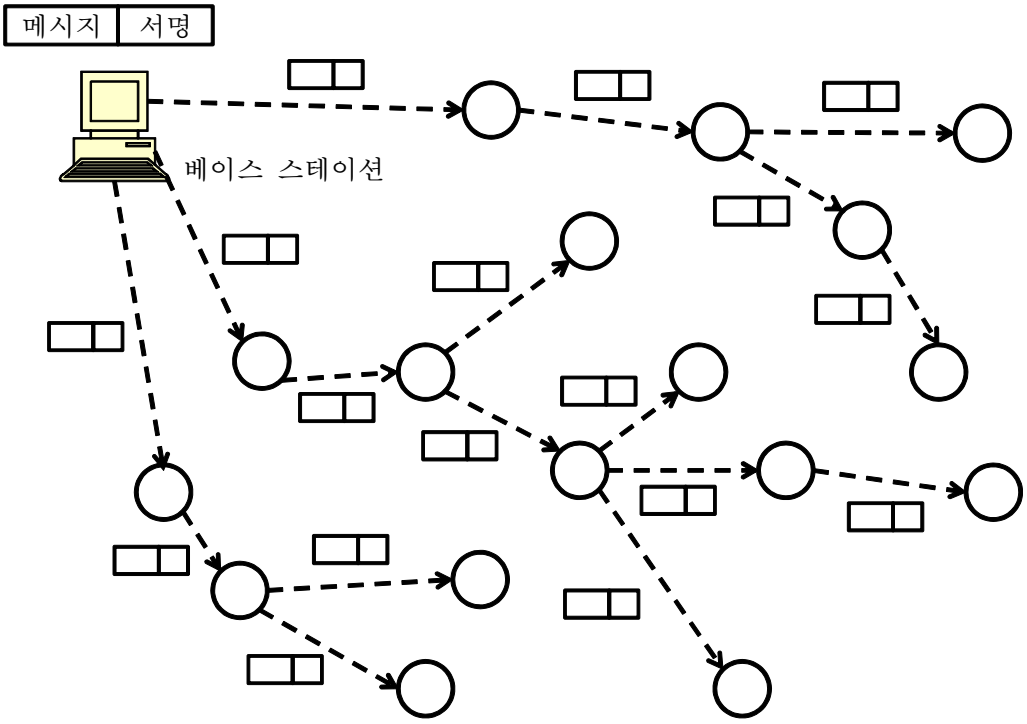

그림 1.1: 센서 네트워크 모델

베이스 스테이션이 센서 노드를 대신해 메시지를 브로드캐스트해주면 된다.

제한된 통신 범위로 인해 네트워크 안의 각 센서 노드들은 그림 1.1에서 볼 수 있듯이 베이스 스테이션으로부터 직접 또는 다른 센서 노드들을 거쳐 브로드캐스트 메시지를 받는다. 브로드캐스트 메시지는 메시지와 서명으로 구성되는데 모든 센서 노드들이 같은 메시지와 서명을 받는다는 것만 보장해주면 본 논문에서 제안하는 일회용 서명 기법은 이 메시지와 서명을 어떠한 경로를 통해 받았는지는 상관하지 않는다.

1.3 논문의 구성

본 논문은 다음과 같이 구성된다. 2장에서는 무선 센서 네트워크에서 브로드캐스트 인증을 수행하기 위해 기존에 제안된 다양한 기법들을 논의한다. 본 논문에서 제안하는 무선 센서 네트워크에서 브로드캐스트 인증을 위한 효율적인 일회용 서명 기법인 HORSIC은 3장에서 설명하며, 4장에서는 HORSIC의 성능을 평가하고 보안성을 분석하며 이를 다양한 일회용 서명 기법들과 비교한다. 마지막으로 5장에서 결론을 맺는다.

제 2 장 관련 연구

본 장에서는 무선 센서 네트워크에서 브로드캐스트 인증을 수행하기 위해 기존에 제안된 다양한 기법들을 논의할 것이다. 2.1절에서는 μTESLA [PST+02]에 대해 살펴보고, 2.2절에서는 Lamport 일회용 서명 기법 [DH76, KL08], Merkle-Winternitz 일회용 서명 기법 [Mer79], BiBa [Per01], HORS [RR02] 등의 다양한 일회용 서명 기법들을 살펴볼 것이다.

본 논문에서 사용하는 용어들의 의미는 다음과 같다. 보안 매개 변수 l이 주어졌을 때, 함수 $f : \{0,1\}^l \rightarrow \{0,1\}^l$는 크기 l인 비트열에서 동작하는 일방향 함수 (one-way function)를 나타낸다. 함수 f가 일방향 함수이면 모든 입력 값 x에 대해 출력 값 $f(x)$를 구하는 다항시간 알고리즘 (polynomial time algorithm)은 존재하지만, $\{0,1\}^l$에서의 균등 분포 (uniform distribution)를 나타내는 확률 변수 (random variable) X에 대해 $f(X)$가 주어졌을 때, $f^{-1}(f(X))$를 구하는 확률적 다항시간 알고리즘 (randomized polynomial time algorithm)은 무시할 수 있을 정도의 작은 확률로만 존재한다 [MVOV97]. 이러한 일방향 함수는 빠르게 계산되는 암호학적 해시 함수 (SHA-1 [NIS02], MD5 [Riv92])나 블록 암호를 통해 구현 가능하다 [MMO85, MOI90, Pre93, PGV94]. $f^k(x)$는 입력 값 x에 함수 f를 k번 적용한 것을 의미하고, $f^0(x) = x$이다.

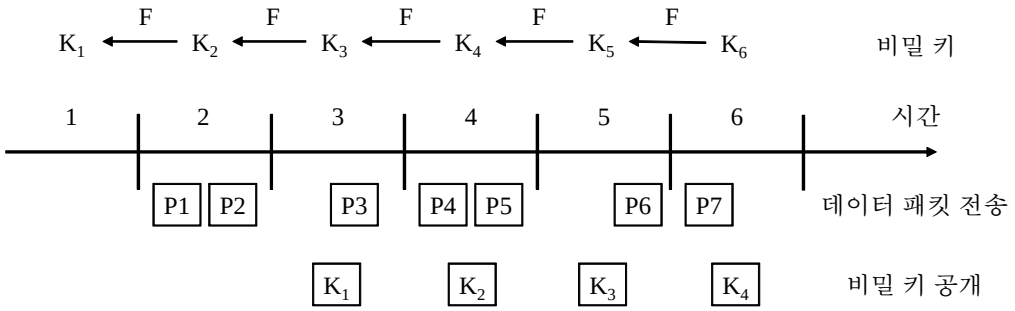

그림 2.1: μTESLA의 인증 과정

2.1 μTESLA

Perrig et al.은 SPINS (Security Protocols for Sensor Networks)라 불리는 제한된 자원을 가진 무선 센서 네트워크를 위한 보안 프로토콜들의 집합을 제안하였다 [PST+02]. SPINS는 SNEP (Secure Network Encryption Protocol)와 μTESLA (the micro version of the Timed, Efficient, Streaming, Loss-tolerant Authentication Protocol)로 이루어지는데 SNEP는 데이터 기밀성, 일대일 통신 인증, 리플레이 공격 방지 등의 기능을 제공하고, μTESLA는 브로드캐스트 인증 기능을 제공한다. 본 절에서는 μTESLA에 대해 살펴본다.

μTESLA는 TESLA [PTSC00, PCST01]를 무선 센서 네트워크 환경에서 사용할 수 있도록 변형한 프로토콜로서 라우팅 정보 [KW03]나 데이터 병합 메시지 [HE03, PSP03]와 같은 다양한 정보들을 인증하는데 사용된다. 브로드캐스트 인증을 위해서는 오직 송신자만 인증 정보를 생성할 수 있고 수신자들은 인증 정보를 검증만 할 수 있도록 하는 비대칭적 요소가 있어야 하는데 μTESLA는 시간을 이러한 비대칭적 요소로 사용한다. 따라서 μTESLA는 송신자와 수신자들 간의 시간 동기화를 필요로 하며 이는 다양한 시간 동기화 기법들을 이용해서 구현할 수 있다 [EGE02, ER03].

그림 2.1은 μTESLA의 인증 과정을 나타낸다. 우선 송신자는 전체 시간을 고정된 시간 간격으로 나눈다. 이 때 나누어진 전체 시간 간격의 수를 N이라 하자. 그리고 나서 송신자는 MAC 생성을 위해 특정 시간 간격에만 사용되는 비밀 키 K_1, K_2, ..., K_N을 생성한다. 그림 2.1에서 볼 수 있듯이, 시간 간격 i에는 비밀 키 K_i만 MAC 생성을 위해 사용할 수 있다.

비밀 키 K_1, K_2, ..., K_N은 다음과 같이 계산된다. 우선 마지막 시간 간격 N에 사용되는 비밀 키 K_N을 랜덤하게 생성한다. 나머지 비밀 키 K_1, K_2, ..., K_{N-1}은 아래 식 2.1과 같이 SHA-1이나 MD5 같은 암호학적 해시 함수로 구현된 일방향 함수 F를 반복적으로 적용함으로써 구할 수 있다.

$$\begin{aligned} K_{N-1} &= F^1(K_N) = F(K_N) \\ K_{N-2} &= F^2(K_N) = F(K_{N-1}) \\ K_{N-3} &= F^3(K_N) = F(K_{N-2}) \\ &\cdots \\ K_1 &= F^{N-1}(K_N) = F(K_2) \end{aligned} \tag{2.1}$$

위의 식 2.1에서 볼 수 있듯이 비밀 키 값들은 사용 순서와 반대로 일방향 키 체인을 이루면서 생성된다. 따라서 특정 시간 간격에 사용되는 비밀 키 값을 알면 그보다 이전 시간 간격들을 위한 비밀 키 값들은 쉽게 구할 수 있다.

데이터를 보낼 때 송신자는 현재 시간 간격을 위한 비밀 키를 사용해서 MAC을 계산하고, 이를 데이터와 함께 수신자에게 전송한다. 그림 2.1을 예로 들면 시간 간격 2에 전송되는 데이터 패킷 P1과 P2는 비밀 키 K_2를 사용해서 계산된 MAC과 함께 보내지고, 시간 간격 3에 전송되는 데이터 패킷 P3는 비밀 키 K_3를 사용해서 계산된

MAC과 함께 보내진다.

이 때 오직 송신자만이 MAC 계산을 위해 사용된 비밀 키 값을 알고 있으므로 수신자는 데이터 패킷과 MAC을 받아도 바로 인증은 불가능하다. 때문에 수신자는 데이터 패킷과 MAC을 받으면 우선 이 MAC이 현재 공개되지 않은 비밀 키를 사용해서 계산된 것인지만 확인하고 데이터 패킷과 함께 버퍼에 저장한다.

어느 정도 미리 정해진 시간이 지난 뒤에 송신자는 MAC 생성을 위해 사용되었던 비밀 키 값을 공개한다. 그림 2.1을 예로 들면 시간 간격 1에 MAC 생성을 위해 사용되었던 비밀 키 K_1은 시간 간격 3에 공개되었고, 시간 간격 2에 MAC 생성을 위해 사용되었던 비밀 키 K_2는 시간 간격 4에 공개되었다. 따라서 그림 2.1의 경우 비밀 키 공개를 위한 지연 시간 간격은 2이다.

여기서 공개되는 비밀 키 값들은 앞의 식 2.1에서 보다시피 일방향 키 체인을 이용해서 만들어지므로 수신자가 새로운 비밀 키를 받으면 그것이 올바른 것인지 쉽게 검증 가능하다. 그러므로 송신자가 새로운 비밀 키를 공개하면 수신자는 우선 공개된 비밀 키가 올바른지 먼저 확인하고, 올바를 경우 MAC 계산을 통해 버퍼에 저장된 데이터 패킷을 인증한다. 그림 2.1을 예로 들면 시간 간격 2에 전송되는 데이터 패킷 P1과 P2는 시간 간격 4에 인증 가능하고, 시간 간격 3에 전송되는 데이터 패킷 P3는 시간 간격 5에 인증 가능하다.

이와 같이 μTESLA는 오직 대칭적인 요소들만을 사용해서 효율적인 브로드캐스트 인증을 수행한다. 하지만 송신자와 수신자들간의 시간 동기화를 필요로 하며 수신자가 데이터 패킷을 받아도 바로 인증하지 못하고 어느 정도 정해진 시간이 지난 후에만 인증이 가능하다는 단점이 있다.

2.2 일회용 서명 기법

무선 센서 네트워크에서 브로드캐스트 인증은 일회용 서명 기법들을 통해서도 가능하다 [DH76, Rab78, Lam79, Mer88, Mer89, EGM89, BM94, BM96, Roh99, Per01, MP02, RR02, BGD+06, PC06, CSLH06, LC11]. 일회용 서명 기법들은 일방향 함수만을 기반으로 하므로 RSA [RSA78] 나 DSA [LG09] 같은 일반적인 공개 키 기반 서명 기법들보다 서명 생성과 검증에 많은 계산을 필요로 하지 않아 무선 센서 네트워크 환경에 적합하다. 또한 μTESLA와는 달리 송신자와 수신자들간의 시간 동기화를 필요로 하지도 않으며, 수신자가 데이터 패킷을 받자마자 바로 인증을 수행할 수도 있다. 더구나 기존의 RSA나 DSA 같은 일반적인 공개 키 기반 서명 기법들은 대부분 소인수 분해 (prime factorization) 나 이산 대수 (discrete logarithms) 문제의 어려움에 기반을 하고 있어 양자 컴퓨터 [NCG02] 가 본격적으로 실용화 된다면 더 이상 사용할 수 없게 되지만 [Sho99], 일방향 함수를 기반으로 하는 일회용 서명 기법들은 여전히 사용 가능하다 [Gro96].

하지만 일회용 서명 기법을 무선 센서 네트워크에서 브로드캐스트 인증을 위해 사용하기 위해서는 해결해야 할 몇 가지 문제가 있다.

- 일회용 서명 기법은 이름 그대로 주어진 비밀 키로 오직 한 번만 서명이 가능하도록 설계된다. 브로드캐스트 인증에 사용하기 위해서는 이러한 일회용 서명 기법을 여러 번 사용할 수 있도록 확장해야 한다.
- 공개 키의 저장을 위해 많은 양의 메모리를 필요로 하며 서명의 크기가 커서 높은 통신 부하를 야기한다. 센서 네트워크에서 사용하기 위해서는 공개 키의 크기를 줄이고 서명의 크기도 줄여야 한다.

본 절에서는 Lamport 일회용 서명 기법 [DH76, KL08], Merkle-Winternitz 일회용 서명 기법 [Mer79], BiBa [Per01], HORS [RR02] 등의 기존에 제안된 다양한 일회용 서명 기법들을 살펴볼 것이다.

2.2.1 Lamport 일회용 서명 기법

한 비트로 된 메시지를 서명하는 경우를 먼저 살펴보자. 송신자는 두 개의 난수 x_0 와 x_1 을 생성하고, 각각에 일방향 함수 f 를 적용해 y_0 와 y_1 을 구한다 ($y_0 = f(x_0)$, $y_1 = f(x_1)$). 이 때 y_0 와 y_1 은 외부에 공개하고 x_0 와 x_1 은 송신자만 알도록 비밀로 한다. 그 후 메시지를 서명할 때, 서명하고자 하는 메시지가 '0' 이면 x_0 를 서명으로 공개하고, '1' 이면 x_1 을 서명으로 공개하면 된다.

이 경우 메시지 값이 '0' 이면 수신자는 x_0 를 제시하고 $y_0 = f(x_0)$ 임을 보임으로써 송신자가 '0' 이라는 메시지에 대해 서명을 했음을 제3자에게 증명할 수 있다. x_0 와 x_1 값은 송신자만이 알고 있고, 공개된 y_0 와 y_1 값만 가지고 $y_0 = f(x'_0)$ 이나 $y_1 = f(x'_1)$ 인 x'_0 나 x'_1 을 구하는 것은 일방향 함수의 특성에 의해 불가능하므로 수신자가 $y_0 = f(x_0)$ 인 x_0 를 제시할 수 있었다는 것은 송신자가 '0' 이라는 메시지에 대해 서명의 의미로 x_0 를 수신자에게 공개했음을 의미하기 때문이다.

위의 알고리즘을 여러 비트로 바로 확장시키면 Lamport 일회용 서명 기법이 된다 [DH76, KL08]. 알고리즘 2.1, 2.2, 2.3은 Lamport 일회용 서명 기법의 키 생성, 서명 생성, 서명 검증 과정을 각각 기술한다.

알고리즘 2.1 키 생성 (Lamport 일회용 서명 기법)

입력: 보안 매개 변수 l, 서명하고자 하는 메시지의 길이 $|m|$

출력: 비밀 키 $\begin{pmatrix} x_{1,0} & x_{2,0} & \dots & x_{|m|,0} \\ x_{1,1} & x_{2,1} & \dots & x_{|m|,1} \end{pmatrix}$, 공개 키 $\begin{pmatrix} y_{1,0} & y_{2,0} & \dots & y_{|m|,0} \\ y_{1,1} & y_{2,1} & \dots & y_{|m|,1} \end{pmatrix}$

1: 모든 $i \in \{1, 2, \dots, |m|\}$에 대해 다음과 같이 난수 생성 $x_{i,0}, x_{i,1} \leftarrow \{0,1\}^l$

2: 모든 $i \in \{1, 2, \dots, |m|\}$에 대해 $y_{i,0} = f(x_{i,0})$, $y_{i,1} = f(x_{i,1})$

알고리즘 2.2 서명 생성 (Lamport 일회용 서명 기법)

입력: 메시지 m, 비밀 키 $\begin{pmatrix} x_{1,0} & x_{2,0} & \dots & x_{|m|,0} \\ x_{1,1} & x_{2,1} & \dots & x_{|m|,1} \end{pmatrix}$

출력: 서명 $(sig_1, sig_2, \dots, sig_{|m|})$

1: 메시지 m을 비트 단위로 나눔 $m = m_1 m_2 \dots m_{|m|}$

2: 모든 $i \in \{1, 2, \dots, |m|\}$에 대해 $sig_i = x_{i,m_i}$

알고리즘 2.3 서명 검증 (Lamport 일회용 서명 기법)

입력: 메시지 m, 서명 $(sig_1, sig_2, \dots, sig_{|m|})$, 공개 키 $\begin{pmatrix} y_{1,0} & y_{2,0} & \dots & y_{|m|,0} \\ y_{1,1} & y_{2,1} & \dots & y_{|m|,1} \end{pmatrix}$

출력: "성공" or "실패"

1: 메시지 m을 비트 단위로 나눔 $m = m_1 m_2 \dots m_{|m|}$

2: 모든 $i \in \{1, 2, \dots, |m|\}$에 대해 $f(sig_i)$와 y_{i,m_i}를 비교. 모두 같은 값이 나오면 "성공", 그렇지 않으면 "실패"

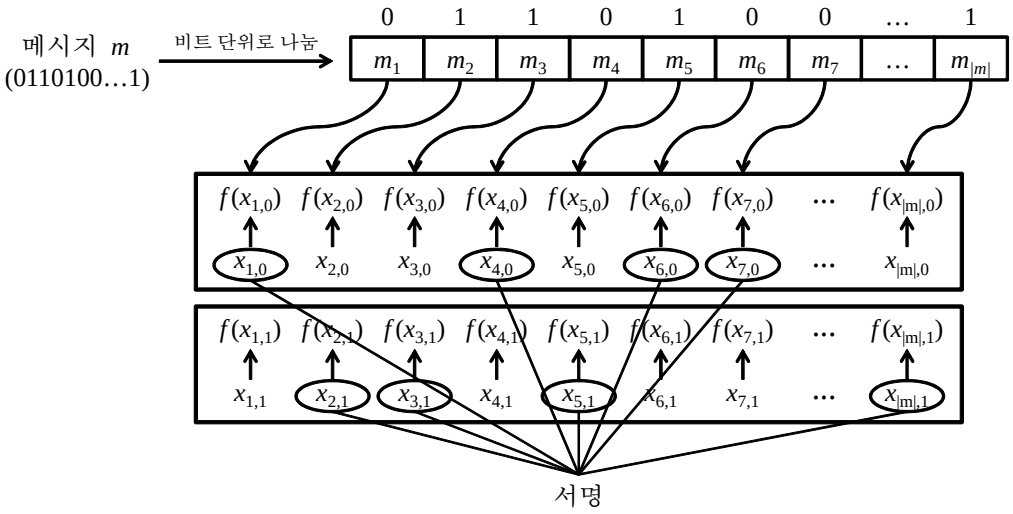

그림 2.2: Lamport 일회용 서명 기법의 서명 생성 과정 ($m = 0110100\ldots1$ 일 때)

그림 2.2는 메시지 $m = 0110100\ldots1$ 일 때 Lamport 일회용 서명 기법의 서명 생성 과정을 보여준다. 그림 2.2를 보면 메시지 m의 첫 번째 비트 m_1의 값이 0이기 때문에 $x_{1,0}$과 $x_{1,1}$ 중 $x_{1,0}$을 서명에 포함시킨 것을 알 수 있다. 마찬가지로 메시지 m의 두 번째 비트 m_2의 값이 1이기 때문에 $x_{2,0}$과 $x_{2,1}$ 중 $x_{2,1}$을 서명에 포함시켰다. 나머지 메시지 비트들도 같은 방법으로 서명을 생성한다.

이러한 Lamport 일회용 서명 기법의 가장 큰 단점은 비밀 키와 공개 키 그리고 서명의 크기가 너무 크다는 것이다. 예를 들어 보안 매개 변수 l의 크기가 80 비트라면 비밀 키와 공개 키의 크기는 메시지 크기의 160 배이고, 서명의 크기는 메시지 크기의 80 배가 된다. 게다가 주어진 비밀 키로 오직 한 번만 서명이 가능하기 때문에 센서 네트워크 환경에서 브로드캐스트 인증을 위해 사용하기에는 부적절하다.

2.2.2 Merkle-Winternitz 일회용 서명 기법

Merkle은 Lamport 일회용 서명 기법에 비해 서명의 크기를 거의 반으로 줄일 수 있는 방법을 제안하였다 [Mer79]. Merkle의 방법에서는 Lamport 일회용 서명 기법에서와는 다르게 각각의 메시지 비트에 대해 x와 y를 두 개씩 유지하는 대신 하나만 유지한다. 그 후 메시지를 서명할 때는 서명하고자 하는 메시지 비트가 '1'일 때만 그에 해당하는 위치의 x를 서명에 포함시키고 '0'일 때는 서명에 포함시키지 않는다. 이 경우 수신자가 실제로는 특정 위치의 x 값을 받았는데 받지 않았다고 주장함으로써 메시지 비트를 '1'에서 '0'으로 바꿀 수가 있는데, 이는 메시지 안에 포함된 '0'의 개수를 2진수로 인코딩하여 메시지 뒷 부분에 붙임으로써 방지할 수 있다.

Winternitz는 메시지에 따라 서로 다른 횟수의 일방향 함수를 서명에 적용시킴으로써 서명의 크기를 줄일 수 있는 방법을 제안하였다 [Mer88, Mer89]. Winternitz의 기본 아이디어를 설명하기 위해 두 비트로 된 메시지를 서명하는 경우를 살펴보자. 송신자는 두 개의 난수 x_0와 x_1을 생성하고, 각각에 일방향 함수 f를 3번 적용해 y_0와 y_1을 구한다 ($y_0 = f^3(x_0)$, $y_1 = f^3(x_1)$). 이 때 Lamport 일회용 서명 기법에서와 마찬가지로 y_0와 y_1은 외부에 공개하고 x_0와 x_1은 송신자만 알도록 비밀로 한다. 메시지에 대한 서명은 다음과 같다.

- 메시지가 '0'일 경우 서명은 $(f^0(x_0), f^3(x_1))$이다.
- 메시지가 '1'일 경우 서명은 $(f^1(x_0), f^2(x_1))$이다.
- 메시지가 '2'일 경우 서명은 $(f^2(x_0), f^1(x_1))$이다.
- 메시지가 '3'일 경우 서명은 $(f^3(x_0), f^0(x_1))$이다.

여기서 x_0와 x_1의 두 개의 값을 사용하는 이유는 x_0만 사용하면 수신자가 실제로는 '1'에 해당하는 서명인 $f^1(x_0)$를 받았는데 여기에 일방향 함수 f를 적용해 '2'에 해당하는 서명인 $f^2(x_0)$를 받았다고 주장할 수도 있기 때문이다. x_1도 같이 사용하면 $f^2(x_1)$을 가지고 $f^1(x_1)$을 구해야 수신자가 위와 같은 주장을 할 수 있는데 이는 일방향 함수의 특성에 의해 불가능하다. 따라서 x_1도 같이 사용함으로써 위와 같은 서명 위조를 막을 수 있다.

이러한 아이디어를 일반화시켜 두 비트가 아닌 $|m|$ 비트로 된 메시지 m을 서명할 때는 공개 키를 생성할 때 일방향 함수 f를 $2^{|m|}-1$번 적용하고 ($y_0 = f^{2^{|m|}-1}(x_0)$, $y_1 = f^{2^{|m|}-1}(x_1)$), 서명으로 ($f^m(x_0)$, $f^{2^{|m|}-1-m}(x_1)$)을 사용하면 된다.

위의 Merkle의 아이디어와 Winternitz의 아이디어를 결합하면 Merkle-Winternitz 일회용 서명 기법이 된다 [Mer88, Mer89]. 알고리즘 2.4, 2.5, 2.6은 Merkle-Winternitz 일회용 서명 기법의 키 생성, 서명 생성, 서명 검증 과정을 각각 기술한다.

알고리즘 2.4 키 생성 (Merkle-Winternitz 일회용 서명 기법)

입력: 보안 매개 변수 l, Winternitz 매개 변수 w, 서명하고자 하는 메시지의 길이 $|m|$

출력: 비밀 키 $(x_1, x_2, \ldots, x_L)$, 공개 키 $(y_1, y_2, \ldots, y_L)$

1: 모든 $i \in \{1, 2, \ldots, L\}$에 대해 다음과 같이 난수 생성 $x_i \leftarrow \{0,1\}^l$

2: 모든 $i \in \{1, 2, \ldots, L\}$에 대해 $y_i = f^{w-1}(x_i)$

$(L = L_1 + L_2,\ L_1 = \lceil \frac{|m|}{\log_2 w} \rceil,\ L_2 = \lfloor \frac{\log_2(L_1(w-1))}{\log_2 w} \rfloor + 1)$

알고리즘 2.5 서명 생성 (Merkle-Winternitz 일회용 서명 기법)

입력: 메시지 m, 비밀 키 $(x_1, x_2, \ldots, x_L)$

출력: 서명 $(sig_1, sig_2, \ldots, sig_L)$

1: 메시지 m을 $\log_2 w$ 비트 단위로 나눔 $m = m_1 m_2 \ldots m_{L_1}$

2: 체크섬 $c = \sum_{i=1}^{L_1}(w - 1 - m_i)$ 계산

3: 체크섬 c를 $\log_2 w$ 비트 단위로 나눔 $c = c_1 c_2 \ldots c_{L_2}$

4: 모든 $i \in \{1, 2, \ldots, L_1\}$에 대해 $sig_i = f^{m_i}(x_i)$

5: 모든 $i \in \{1, 2, \ldots, L_2\}$에 대해 $sig_{(L_1+i)} = f^{c_i}(x_{(L_1+i)})$

알고리즘 2.6 서명 검증 (Merkle-Winternitz 일회용 서명 기법)

입력: 메시지 m, 서명 $(sig_1, sig_2, \ldots, sig_L)$, 공개 키 $(y_1, y_2, \ldots, y_L)$

출력: "성공" or "실패"

1: 메시지 m을 $\log_2 w$ 비트 단위로 나눔 $m = m_1 m_2 \ldots m_{L_1}$

2: 체크섬 $c = \sum_{i=1}^{L_1}(w - 1 - m_i)$ 계산

3: 체크섬 c를 $\log_2 w$ 비트 단위로 나눔 $c = c_1 c_2 \ldots c_{L_2}$

4: 모든 $i \in \{1, 2, \ldots, L_1\}$에 대해 $f^{w-1-m_i}(sig_i)$ 와 y_i를 비교

5: 모든 $i \in \{1, 2, \ldots, L_2\}$에 대해 $f^{w-1-c_i}(sig_{(L_1+i)})$와 $y_{(L_1+i)}$를 비교

6: 위의 두 결과, 모두 같은 값이 나오면 "성공", 그렇지 않으면 "실패"

그림 2.3은 메시지 $m = 101010100\ldots111$ 일 때 Merkle-Winternitz 일회용 서명 기법의 서명 생성 과정을 보여준다. 그림 2.3에서 Winternitz 매개 변수 w 가 8 이라면 $\log_2 w = 3$ 이므로 메시지 m 을 3 비트 단위로 나눈다. 이 경우 메시지 m 의 첫 번째 세 비트 m_1 의 값은 101 로서 이 값을 이진수로 해석하면 5 가 되므로 $f^5(x_1)$ 이 서명에 포함된다. 마찬가지로 메시지 m 의 두 번째 세 비트 m_2 의 값은 010 으로서 이 값을 이진수로 해석하면 2 가 되므로 $f^2(x_2)$ 이 서명에 포함된다. 나머지 메시지 비트들도 같은 방법으로 서명을 생성한다. 전체 비밀 키 $(x_1, x_2, \ldots, x_L)$ 중에 $(x_1, x_2, \ldots, x_{L_1})$ 은 앞에서 설명한 바와 같이 메시지를 표현하기 위해 사용되고, 나머지 $(x_{L_1+1}, x_{L_1+2}, \ldots, x_L)$ 은 체크섬을 표현하기 위해 사용된다.

이 기법은 의사 난수 함수들의 패밀리 (a family of pseudo random functions) 를 통해 구현될 경우 적응적 선택 메시지 공격 (adaptive chosen-message attack) 에 대해 존재적 위조불가 (existentially unforgeable) 임이 증명되었다 [BDE+11].

Merkle-Winternitz 일회용 서명 기법에서 Winternitz 매개 변수 w 는 서명 생성 및 검증 비용과 서명 크기 사이의 트레이드 오프 (trade-off) 를 가능하게 한다. w 값이 커지면 서명 생성 및 검증 비용은 늘어나지만 서명 크기는 줄어든다. 하지만 적절한 w 값을 통해 서명 크기를 줄이더라도 센서 네트워크 환경에서 사용하기에 서명 크기는 여전히 큰 편이다. 게다가 주어진 공개 키로 오직 한 번만 서명과 검증이 가능하도록 설계되었기 때문에 여러 번 사용할 수 있도록 확장해야 하는데, Merkle 트리 기법 [Mer89] 과 결합하는 방법이 있기는 하지만, 이 경우 거의 무제한으로 서명이 가능해지는 대신 서명 크기가 꽤 증가하게 된다 [BDK+07].

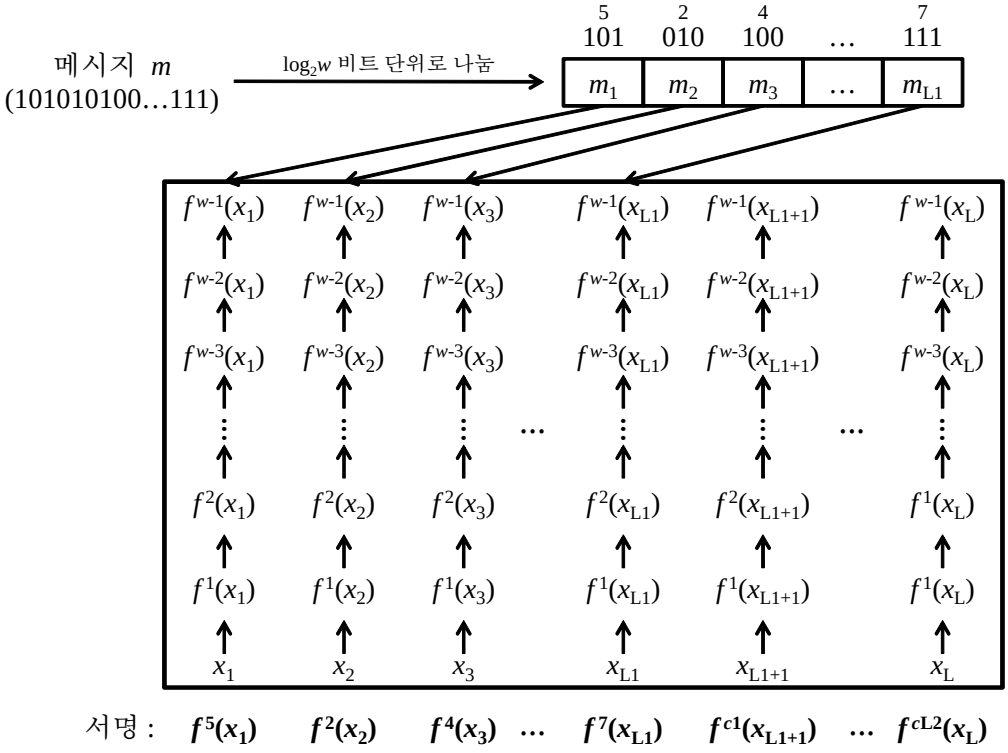

그림 2.3: Merkle-Winternitz 일회용 서명 기법의 서명 생성 과정

2.2.3 BiBa

Perrig은 BiBa (Bins and Balls) 라 불리는 효율적인 일회용 서명 기법을 제안하였다 [Per01]. BiBa는 생일 패러독스 (birthday paradox) 를 이용해 기존의 일회용 서명 기법들에 비해 효율성과 보안성을 높인다.

BiBa에서 송신자는 t 개의 난수 값을 생성하여 이를 비밀 키로 하고, 각각의 비밀 키에 서약 값 (commitment) 으로서 일방향 함수 f 를 적용해 이를 공개 키로 한다. 서명 생성시 각 비밀 키 값은 메시지 m 에 따라 선택된 해시 함수 G_h 에 의해 0 부터 $n-1$ 사이의 한 수에 대응되는데, 그 중 k 개 이상의 서로 다른 비밀 키 값이 해시 함수 G_h 에 의해 같은 값에 대응되면 이 비밀 키 값들이 서명이 된다.

BiBa에서 송신자는 t 개의 비밀 키 값들을 가지고 있으므로 생일 패러독스를 이용해 서명을 쉽게 생성할 수 있지만, 공격자는 가지고 있는 비밀 키 값이 적으므로 서명을 위조하기가 쉽지 않다. 알고리즘 2.7, 2.8, 2.9는 BiBa의 키 생성, 서명 생성, 서명 검증 과정을 각각 기술한다.

알고리즘 2.7 키 생성 (BiBa)

입력: 보안 매개 변수 l, t

출력: 비밀 키 $(s_1, s_2, \ldots, s_t)$, 공개 키 $(v_1, v_2, \ldots, v_t)$

1: 모든 $i \in \{1, 2, \ldots, t\}$ 에 대해 다음과 같이 난수 생성 $s_i \leftarrow \{0,1\}^l$

2: 모든 $i \in \{1, 2, \ldots, t\}$ 에 대해 $v_i = f(s_i)$

알고리즘 2.8 서명 생성 (BiBa)

입력: 메시지 m, 비밀 키 $(s_1, s_2, \ldots, s_t)$

출력: 서명 $(c, sig_1, sig_2, \ldots, sig_k)$

1: $c = 0$

2: 다음과 같이 해시 값을 계산 $h = H(m|c)$

3: 해시 함수 패밀리 G로부터 h를 사용해서 하나의 해시 함수 G_h를 선택

4: 모든 $i \in \{1, 2, \ldots, t\}$에 대해 $G_h(s_i)$를 계산하고, 같은 $G_h(s_i)$ 값을 생성하는 k개의 서로 다른 s_i가 존재하는지 확인

($s_{\alpha_1} \neq s_{\alpha_2} \neq \cdots \neq s_{\alpha_k}$ 이고 $G_h(s_{\alpha_1}) = G_h(s_{\alpha_2}) = \cdots = G_h(s_{\alpha_k})$ 이어야 함)

5: 존재하면, 모든 $j \in \{1, 2, \ldots, k\}$에 대해 $sig_j = s_{\alpha_j}$

6: 존재하지 않으면, c를 1 증가시키고 2번으로 되돌아감

알고리즘 2.9 서명 검증 (BiBa)

입력: 메시지 m, 서명 $(c, sig_1, sig_2, \ldots, sig_k)$, 공개 키 $(v_1, v_2, \ldots, v_t)$

출력: "성공" or "실패"

1: 다음과 같이 해시 값을 계산 $h = H(m|c)$

2: 해시 함수 패밀리 G로부터 h를 사용해서 하나의 해시 함수 G_h를 선택

3: 만약 $p \neq q$이고 $sig_p = sig_q$인 p와 q가 존재하면 $(p, q \in \{1, 2, \ldots, k\})$, "실패"

4: 모든 $j \in \{1, 2, \ldots, k\}$에 대해 $f(sig_j)$와 같은 값을 가진 v_i가 $i \in \{1, 2, \ldots, t\}$에 있는지 확인. 하나라도 없으면 "실패"

5: 모든 $j \in \{1, 2, \ldots, k\}$에 대해 $G_h(sig_j)$를 계산하고 모두 같은 값이 나오는지 확인. 모두 같은 값이 나오면 "성공", 그렇지 않으면 "실패"

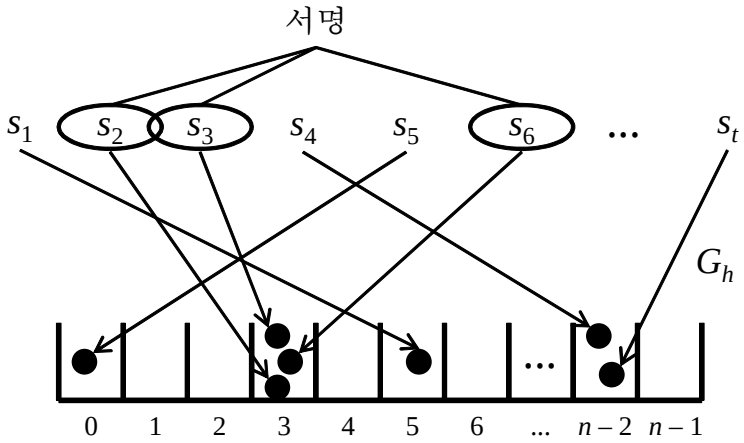

그림 2.4: BiBa의 서명 생성 과정 ($k = 3$일 때)

그림 2.4는 $k = 3$일 때 BiBa의 서명 생성 과정을 보여준다. 여기서 s_1, s_2, ... , s_t는 비밀 키 값들이며, 각 비밀 키 값은 메시지 m에 따라 선택된 해시 함수 G_h에 의해 0부터 $n-1$ 사이의 한 수에 대응된다. 그림 2.4의 경우, s_2, s_3, s_6이 같은 값에 대응이 되므로 이 값들이 서명이 된다.

BiBa는 기존의 일회용 서명 기법들에 비해 서명 크기가 작고 검증 연산량이 작은 편이다. 하지만 서명을 생성하는데 너무 오랜 시간을 필요로 하고, 센서 네트워크에서 사용하기에는 공개 키의 크기와 서명의 크기가 여전히 크다.

2.2.4 HORS

Reizin은 HORS (Hash to Obtain Random Subset)라 불리는 효율적인 일회용 서명 기법을 제안하였다 [RR02]. HORS는 t개의 원소를 가지는 비밀 키 집합에서 암호학적 해시 함수 H를 사용해 메시지에 따라 서로 다른 k개의 원소를 선택해 이를 서명에 포함시킨다. 알고리즘 2.10, 2.11, 2.12는 HORS의 키 생성, 서명 생성, 서명 검증 과정을 각각 기술한다.

알고리즘 2.10 키 생성 (HORS)

입력: 보안 매개 변수 l, t

출력: 비밀 키 $(s_1, s_2, \ldots, s_t)$, 공개 키 $(v_1, v_2, \ldots, v_t)$

1: 모든 $i \in \{1, 2, \ldots, t\}$에 대해 다음과 같이 난수 생성 $s_i \leftarrow \{0,1\}^l$

2: 모든 $i \in \{1, 2, \ldots, t\}$에 대해 $v_i = f(s_i)$

알고리즘 2.11 서명 생성 (HORS)

입력: 메시지 m, 비밀 키 $(s_1, s_2, \ldots, s_t)$

출력: 서명 $(sig_1, sig_2, \ldots, sig_k)$

1: 다음과 같이 해시 값을 계산 $h = H(m)$

2: h를 각각의 조각이 $\log_2 t$ 비트가 되도록 k개의 조각으로 분해 $h = h_1 h_2 \ldots h_k$

3: 모든 $j \in \{1, 2, \ldots, k\}$에 대해 h_j를 정수 i_j로 해석

4: 모든 $j \in \{1, 2, \ldots, k\}$에 대해 $sig_j = s_{i_j}$

알고리즘 2.12 서명 검증 (HORS)

입력: 메시지 m, 서명 $(sig_1, sig_2, \ldots, sig_k)$, 공개 키 $(v_1, v_2, \ldots, v_t)$

출력: "성공" or "실패"

1: 다음과 같이 해시 값을 계산 $h = H(m)$

2: h를 각각의 조각이 $\log_2 t$ 비트가 되도록 k개의 조각으로 분해 $h = h_1 h_2 \ldots h_k$

3: 모든 $j \in \{1, 2, \ldots, k\}$에 대해 h_j를 정수 i_j로 해석

4: 모든 $j \in \{1, 2, \ldots, k\}$에 대해 $f(sig_j)$와 v_{i_j}를 비교. 모두 같은 값이 나오면 "성공", 그렇지 않으면 "실패"

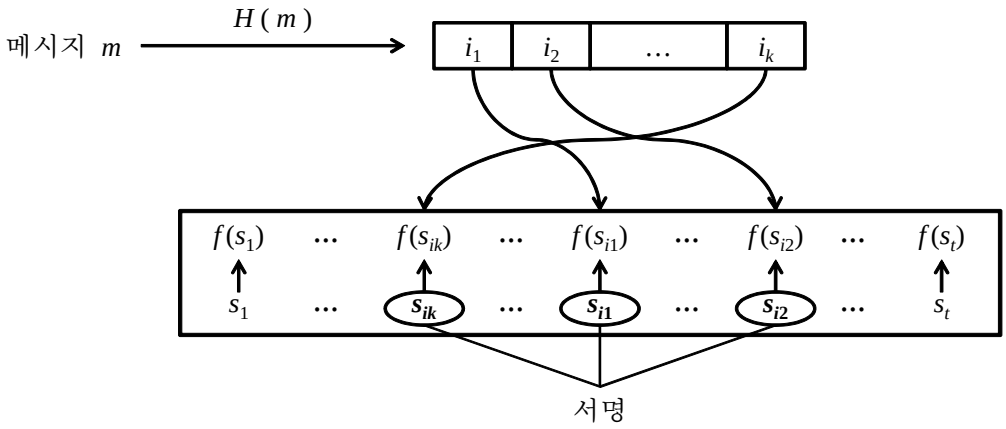

그림 2.5: HORS의 서명 생성 과정

그림 2.5는 HORS의 서명 생성 과정을 보여준다. 여기서 메시지 m은 암호학적 해시 함수 H를 통해 t개의 원소를 가지는 비밀 키 집합 $\{s_1, s_2, \ldots, s_t\}$에서 최대 k개의 비밀 키를 선택해 서명을 생성하는데 사용된다.

HORS에서의 서명 위조 확률은 최대 $\frac{k^k}{t^k}$이다. 그러므로 같은 보안 수준에서 t 값 (공개 키 크기)과 k 값 (서명 크기)을 동시에 줄일 수는 없다. 4.3장에서 보겠지만 HORS는 현재 센서 네트워크 환경에서 사용하기에 적합한 공개 키 크기와 서명 크기를 둘 다 제공하지는 못한다.

제 3 장 브로드캐스트 인증을 위한 효율적인 일회용 서명 기법

본 장에서는 HORSIC이라 부르는 무선 센서 네트워크에서 브로드캐스트 인증을 위한 효율적인 일회용 서명 기법을 제안한다 [LKC+12]. HORSIC은 기본적으로 2.2.4절에서 설명한 HORS [RR02]의 확장이다. HORS는 t개의 원소를 가지는 비밀 키 집합에서 암호학적 해시 함수 H를 사용해 서명하고자 하는 메시지에 따라 서로 다른 최대 k개의 원소를 선택하여 이를 서명에 포함시킨다. HORSIC도 HORS와 같은 이유로 암호학적 해시 함수 H를 사용하지만, 비밀 키 자체를 서명에 포함시키는 대신 또 다른 암호학적 해시 함수 G와 전단사 함수 $C_{k,z}$를 통해 일방향 함수를 서명하고자 하는 메시지에 따라 서로 다른 횟수로 적용하고 난 후에 이를 서명에 포함시킨다.

3.1절에서는 HORSIC에서 서명 생성과 검증을 위해 필요로 하는 함수들을 설명한다. 하나의 메시지만 서명 가능한 HORSIC은 3.2절에서 설명하고, 3.3절에서 이를 다수의 메시지를 서명할 수 있도록 확장한다. 마지막으로 3.4절에서 HORSIC의 동작 과정을 구체적인 예와 함께 설명한다.

3.1 함수 설명

HORSIC은 서명의 생성과 검증을 위해 H와 G 그리고 $C_{k,z}$라는 함수를 사용한다. 보안 매개 변수 t와 k 그리고 z에 대해 $H : \{0,1\}^* \to \{0,1\}^{k \log_2 t}$와 $G : \{0,1\}^* \to [0, \binom{z-1}{k-1})$는 랜덤 오라클 모델 (random oracle model) [BR93]에 기반한 암호학적 해시 함수 (cryptographic hash function)이고, $C_{k,z}$는 $0 \leq g < \binom{z-1}{k-1}$인 g를 입력으로 받아 아래 식 3.1의 g번째 해를 출력하는 전단사 함수 (bijective function)이다.

$$z = \sum_{i=1}^{k} a_i, \ a_i\text{는 1보다 크거나 같은 정수} \tag{3.1}$$

여기서 위의 식 3.1의 해의 개수는 z를 정확히 k개의 부분으로 나누는 경우의 수로서 이항 계수 $\binom{z-1}{k-1}$이다 [And98]. 예를 들어 $k = 3$이고 $z = 5$라 하면 수식 $a_1 + a_2 + a_3 = 5$는 아래 식 3.2와 같이 6개의 해를 가진다 ($\binom{z-1}{k-1} = \binom{5-1}{3-1} = \binom{4}{2} = 6$).

$$\begin{aligned} C_{3,5}(0) &= (1,1,3) \\ C_{3,5}(1) &= (1,2,2) \\ C_{3,5}(2) &= (1,3,1) \\ C_{3,5}(3) &= (2,1,2) \\ C_{3,5}(4) &= (2,2,1) \\ C_{3,5}(5) &= (3,1,1) \end{aligned} \tag{3.2}$$

알고리즘 3.1은 전단사 함수 $C_{k,z}$의 한 구현 예를 보이고 있다. 이 알고리즘은 다음 식 3.3을 기반으로 설계되었다.

$$\begin{aligned}\binom{z-1}{k-1} &= \binom{z-2}{k-2} + \binom{z-3}{k-2} + \cdots + \binom{k-2}{k-2} \\ &= \sum_{j=1}^{z-k+1} \binom{z-1-j}{k-2}\end{aligned} \tag{3.3}$$

식 3.3의 좌변에 있는 $\binom{z-1}{k-1}$은 식 3.1의 전체 해의 개수를 나타낸다. 식 3.3의 우변에 있는 $\binom{z-2}{k-2}$는 $a_1 = 1$일 때 식 3.1의 해의 개수이고, $\binom{z-3}{k-2}$는 $a_1 = 2$일 때 식 3.1의 해의 개수이다. 식 3.1이 해를 가지기 위해서 a_1이 가질 수 있는 값은 1부터 $z-k+1$까지이므로 식 3.3은 식 3.1의 전체 해의 개수를 a_1의 값에 따라 나눈다.

위의 식 3.3을 바탕으로 알고리즘 3.1의 동작 과정을 자세히 살펴보면 다음과 같다. $a_1 = 1$일 때 식 3.1의 해의 개수는 $\binom{z-2}{k-2}$이므로, 만약 g 값이 $\binom{z-2}{k-2}$보다 작으면 $a_1 = 1$로 설정하고 식 3.1을 $z - 1 = \sum_{i=2}^{k} a_i$로 바꾼 뒤 나머지 값들을 재귀적으로 계산한다. 만약 g 값이 $\binom{z-2}{k-2}$보다 크거나 같으면 다음으로 g 값이 $\binom{z-2}{k-2} + \binom{z-3}{k-2}$보다 작은지 확인한다. 만약 작으면 $a_1 = 2$로 설정하고 앞에서와 마찬가지로 식 3.1을 $z - 2 = \sum_{i=2}^{k} a_i$로 바꾼 뒤 나머지 값들을 재귀적으로 계산한다. 나머지 경우에도 같은 방법을 사용하면 된다.

알고리즘 3.1에서 조합 연산 (combinatorial operation)을 수행하는 횟수의 느슨한 상계 (loose upper bound)는 $O(kz)$이다. 따라서 알고리즘 3.1은 효율적으로 구현 가능하다.

알고리즘 3.1 전단사 함수 $C_{k,z}(g)$의 한 구현 예

입력: $k \le z$이고 $0 \le g < \binom{z-1}{k-1}$인 k, z, g

출력: $(a_1, a_2, \ldots, a_k)$

$s = 0$
$r = k$
for $i = 1$ to $k - 2$ **do**
    for $j = 1$ to $z - r + 1$ **do**
        if $g < s + \binom{z-1-j}{r-2}$ **then**
            $a_i = j$
            $r = r - 1$
            $z = z - j$
            break ▷ 안쪽 for 루프를 탈출
        end if
        $s = s + \binom{z-1-j}{r-2}$
    end for
end for
$a_{k-1} = g - s + 1$
$a_k = z - a_{k-1}$

3.2 하나의 메시지를 위한 HORSIC

본 절에서는 우선 하나의 메시지만 서명 가능한 HORSIC을 설명한다. 이것을 다수의 메시지를 서명할 수 있도록 확장하는 것은 3.3절에서 다룰 것이다. HORSIC은 키 생성과 서명 생성 그리고 서명 검증으로 이루어진다.

3.2.1 키 생성

알고리즘 3.2는 하나의 메시지를 위한 HORSIC의 키 생성 과정을 보여준다. 우선 t 개의 l 비트 난수 $(s_1, s_2, \ldots, s_t)$를 생성하고, 각각의 s_i에 대해 일방향 함수 f를 w 번 적용해 길이 w의 일방향 체인을 생성한다 $(s_i \to f(s_i) \to \cdots \to f^w(s_i))$. 식 3.1을 만족시키는 모든 a_i 값은 1부터 $z-k+1$까지의 수만 가질 수 있으므로 하나의 메시지를 위한 HORSIC의 경우 w 값은 $z-k+1$로 고정된다.

알고리즘 3.2 키 생성 (하나의 메시지를 위한 HORSIC)

입력: 보안 매개 변수 l, t

출력: 비밀 키 $(s_i, f(s_i), \cdots, f^{w-1}(s_i))$ (모든 $i \in \{1, 2, \ldots, t\}$), 공개 키 $(v_1, v_2, \ldots, v_t)$

1: 모든 $i \in \{1, 2, \ldots, t\}$에 대해 다음과 같이 난수 생성 $s_i \leftarrow \{0,1\}^l$

2: 각각의 s_i에 대해 길이 w의 일방향 체인 생성 $s_i \to f(s_i) \to \cdots \to f^w(s_i)$

3: 모든 $i \in \{1, 2, \ldots, t\}$에 대해 $v_i = f^w(s_i)$

3.2.2 서명 생성

HORSIC은 서명 생성을 위해 3.1절에서 설명한 H와 G 그리고 $C_{k,z}$ 함수를 사용한다. 암호학적 해시 함수 $H : \{0,1\}^* \rightarrow \{0,1\}^{k \log_2 t}$는 t개의 원소를 가지는 비밀 키 집합에서 서명하고자 하는 메시지에 따라 서로 다른 k개의 원소를 선택하는데 사용된다. 또 다른 암호학적 해시 함수 $G : \{0,1\}^* \rightarrow [0, \binom{z-1}{k-1})$와 전단사 함수 $C_{k,z}$는 같이 사용되어 메시지 m을 z의 k-부분 정수 분해인 $(a_1, a_2, \ldots, a_k)$와 대응시킴으로써, 해시 함수 H를 통해 선택된 k개의 원소들에게 메시지에 따라 서로 다른 횟수로 일방향 함수가 적용되도록 한다.

알고리즘 3.3은 하나의 메시지를 위한 HORSIC의 서명 생성 과정을 보여준다. 여기서 카운터 c는 모든 i_j가 서로 다르도록 보장하기 위해 사용된다.

알고리즘 3.3 서명 생성 (하나의 메시지를 위한 HORSIC)

입력: 메시지 m, 비밀 키 $(s_i, f(s_i), \cdots, f^{w-1}(s_i))$ (모든 $i \in \{1, 2, \ldots, t\}$)

출력: 서명 $(c, sig_1, sig_2, \ldots, sig_k)$

1: $g = G(m)$

2: $(a_1, a_2, \ldots, a_k) = C_{k,z}(g)$

3: $c = 0$

4: $h = H(m|c)$

5: h를 각각의 조각이 $\log_2 t$ 비트가 되도록 k개의 조각으로 분해 $h = h_1 h_2 \ldots h_k$

6: 모든 $j \in \{1, 2, \ldots, k\}$에 대해 h_j를 정수 i_j로 해석

7: 만약 $p \neq q$이고 $i_p = i_q$인 p와 q가 존재하면 $(p, q \in \{1, 2, \ldots, k\})$, c를 1 증가시키고 4번으로 되돌아감

8: 모든 $j \in \{1, 2, \ldots, k\}$에 대해 $sig_j = f^{w-a_j}(s_{i_j})$

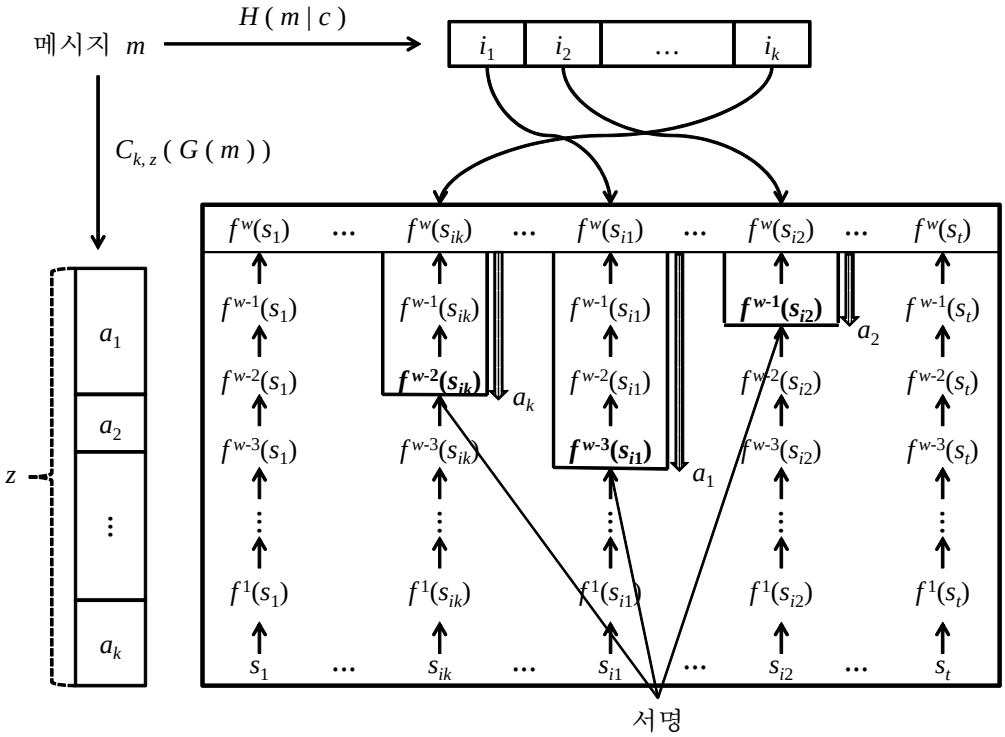

그림 3.1: HORSIC의 서명 생성 과정

그림 3.1은 HORSIC의 서명 생성 과정을 그림으로 표현한 것이다. 메시지 m은 카운터 c와 함께 암호학적 해시 함수 H를 통해 t개의 원소를 가지는 비밀 키 집합 $s_1, s_2, \ldots, s_t$에서 서로 다른 k개의 원소가 선택되도록 돕고, 또 다른 암호학적 해시 함수 G와 전단사 함수 $C_{k,z}$를 통해 z의 k-부분 정수 분해인 $(a_1, a_2, \ldots, a_k)$와 대응됨으로써 함수 H를 통해 선택된 k개의 원소들에게 메시지에 따라 서로 다른 횟수로 일방향 함수가 적용되도록 한다.

3.2.3 서명 검증

알고리즘 3.4는 하나의 메시지를 위한 HORSIC의 서명 검증 과정을 보여준다. HORSIC은 서명 검증을 위해 서명 생성에서와 같은 이유로 H와 G 그리고 $C_{k,z}$ 함수를 사용한다. 함수 H를 통해 선택된 각각의 s_{i_j}에 대해 서명 생성 시 일방향 함수가 $w - a_j$ 번 적용되고 서명 검증 시 일방향 함수가 a_j 번 적용되므로, 서명 생성과 검증이 올바르게 이루어진다면 아래의 알고리즘 3.4는 "성공"을 출력한다.

알고리즘 3.4 서명 검증 (하나의 메시지를 위한 HORSIC)

입력: 메시지 m, 서명 $(c, sig_1, sig_2, \ldots, sig_k)$, 공개 키 $(v_1, v_2, \ldots, v_t)$

출력: "성공" or "실패"

1: $g = G(m)$

2: $(a_1, a_2, \ldots, a_k) = C_{k,z}(g)$

3: $h = H(m|c)$

4: h를 각각의 조각이 $\log_2 t$ 비트가 되도록 k개의 조각으로 분해 $h = h_1 h_2 \ldots h_k$

5: 모든 $j \in \{1, 2, \ldots, k\}$에 대해 h_j를 정수 i_j로 해석

6: 만약 $p \neq q$이고 $i_p = i_q$인 p와 q가 존재하면 $(p, q \in \{1, 2, \ldots, k\})$, "실패"

7: 만약 $f^{a_j}(sig_j) \neq v_{i_j}$ 인 $j \in \{1, 2, \ldots, k\}$가 존재해도 "실패"

8: "성공"

3.3 다수의 메시지들을 위한 HORSIC

본 절에서는 앞의 3.2절에서 설명한 하나의 메시지만 서명 가능한 HORSIC을 다수의 메시지를 서명할 수 있도록 확장한 기법을 설명한다. 이것도 앞에서와 마찬가지로 키 생성과 서명 생성 그리고 서명 검증으로 이루어진다.

3.3.1 키 생성

알고리즘 3.5는 다수의 메시지들을 위한 HORSIC의 키 생성 과정을 보여준다. 우선 t 개의 l 비트 난수 $(s_1, s_2, \ldots, s_t)$를 생성하고, 각각의 s_i에 대해 일방향 함수 f를 w 번 적용해 길이 w의 일방향 체인을 생성한다 $(s_i \rightarrow f(s_i) \rightarrow \cdots \rightarrow f^w(s_i))$. 하나의 메시지를 위한 HORSIC의 키 생성을 기술한 알고리즘 3.2와 다른 점은 w 값이 고정되어 있지 않고 매개 변수로 주어진다는 것 뿐이다.

알고리즘 3.5에서 비밀 키는 $[\{s_1, f(s_1), \ldots, f^{w-1}(s_1)\}, \{s_2, f(s_2), \ldots, f^{w-1}(s_2)\}, \ldots, \{s_t, f(s_t), \ldots, f^{w-1}(s_t)\}]$ 이다. 하지만 비밀 키를 $(s_1, s_2, \ldots, s_t)$로 짧게 줄일 수도 있다. 하지만 이 경우 비밀 키의 크기가 줄어드는 대신 서명 생성 비용이 증가하게 된다. 이 문제에 관해서는 4.1.4절에서 자세히 살펴볼 것이다.

알고리즘 3.5 키 생성 (다수의 메시지들을 위한 HORSIC)

입력: 보안 매개 변수 l, t, w

출력: 비밀 키 $(s_i, f(s_i), \cdots, f^{w-1}(s_i))$ (모든 $i \in \{1, 2, \ldots, t\}$), 공개 키 $(v_1, v_2, \ldots, v_t)$

1: 모든 $i \in \{1, 2, \ldots, t\}$에 대해 다음과 같이 난수 생성 $s_i \leftarrow \{0,1\}^l$

2: 각각의 s_i에 대해 길이 w의 일방향 체인 생성 $s_i \rightarrow f(s_i) \rightarrow \cdots \rightarrow f^w(s_i)$

3: 모든 $i \in \{1, 2, \ldots, t\}$에 대해 $v_i = f^w(s_i)$

일회용 서명에서 서명을 위해 한 번 드러난 키는 다시는 사용될 수가 없다. 따라서 다수의 메시지들을 위한 HORSIC의 경우 하나의 메시지만 서명 가능한 HORSIC과는 다르게 서명을 생성하는 송신자와 서명을 검증하는 수신자가 그들 고유의 내부 상태 정보를 유지해야 한다. 다수의 메시지들을 위한 HORSIC에서 송신자는 키 경계 지수 (key boundary index)라 불리는 내부 상태 변수를 사용하고, 수신자는 현재 공개 키 (current public key)라 불리는 내부 상태 변수를 사용한다. 키 경계 지수는 현재 드러난 일방향 체인 값의 개수를 나타내고, 현재 공개 키는 가장 최근에 드러난 일방향 체인 값을 저장한다.

알고리즘 3.6과 3.7은 다수의 메시지를 위한 HORSIC의 송신자와 수신자 초기화 과정을 각각 보여준다. 송신자는 키 경계 지수 값을 모두 0으로 초기화하고, 수신자는 현재 공개 키 값을 처음에 주어진 공개 키 값으로 초기화한다.

알고리즘 3.6 송신자 초기화 (다수의 메시지들을 위한 HORSIC)

내부 상태 변수: 키 경계 지수 $(b_1, b_2, \ldots, b_t)$

1: 키 경계 지수를 초기화한다: 모든 $i \in \{1, 2, \ldots, t\}$에 대해 $b_i = 0$

알고리즘 3.7 수신자 초기화 (다수의 메시지들을 위한 HORSIC)

내부 상태 변수: 현재 공개 키 $(u_1, u_2, \ldots, u_t)$

입력: 공개 키 $(v_1, v_2, \ldots, v_t)$

1: 현재 공개 키를 초기화한다: 모든 $i \in \{1, 2, \ldots, t\}$에 대해 $u_i = v_i$

3.3.2 서명 생성

알고리즘 3.8은 다수의 메시지들을 위한 HORSIC의 서명 생성 과정을 보여준다. 다수의 메시지들을 위한 HORSIC의 경우에도 하나의 메시지만 서명 가능한 HORSIC과 같은 이유로 H와 G 그리고 $C_{k,z}$ 함수를 사용한다. 알고리즘 3.5를 통해 생성된 일방향 체인 값들의 경우 메시지 서명을 위해 한 번 드러나면 더 이상 사용되어서는 안되므로 송신자는 드러난 일방향 체인 값들에 대한 정보를 유지해야 한다. 이를 위해 현재 드러난 일방향 체인 값의 개수를 나타내는 키 경계 지수를 사용하며, 키 경계 지수는 매번 서명 생성 시 마다 업데이트된다.

알고리즘 3.8을 통해 생성되는 서명들의 경우 패킷 로스를 가정하고 있지 않다. 하지만 키 경계 지수 값을 서명에 포함시킴으로써 패킷 로스가 있더라도 서명 검증이 가능하도록 할 수도 있다 [Per01]. 하지만 이 경우 패킷 로스에 강인해지는 대신 서명의 크기가 증가한다.

알고리즘 3.8 서명 생성 (다수의 메시지들을 위한 HORSIC)

내부 상태 변수: 키 경계 지수 $(b_1, b_2, \ldots, b_t)$

입력: 메시지 m, 비밀 키 $(s_i,\ f(s_i),\ \cdots,\ f^{w-1}(s_i))$ (모든 $i \in \{1, 2, \ldots, t\}$)

출력: 서명 $(c, sig_1, sig_2, \ldots, sig_k)$

1: $g = G(m)$

2: $(a_1, a_2, \ldots, a_k) = C_{k,z}(g)$

3: $c = 0$

4: $h = H(m|c)$

5: h를 각각의 조각이 $\log_2 t$ 비트가 되도록 k개의 조각으로 분해 $h = h_1 h_2 \ldots h_k$

6: 모든 $j \in \{1, 2, \ldots, k\}$에 대해 h_j를 정수 i_j로 해석

7: 만약 $p \neq q$이고 $i_p = i_q$인 p와 q가 존재하면 $(p, q \in \{1, 2, \ldots, k\})$, c를 1 증가시키고 4번으로 되돌아감

8: 모든 $j \in \{1, 2, \ldots, k\}$에 대해 $b_{i_j} = b_{i_j} + a_j$

9: 모든 $j \in \{1, 2, \ldots, k\}$에 대해 $sig_j = f^{w-b_{i_j}}(s_{i_j})$

3.3.3 서명 검증

알고리즘 3.9는 다수의 메시지들을 위한 HORSIC의 서명 검증 과정을 보여준다. 수신자는 서명 검증을 위해 가장 최근에 드러난 일방향 체인 값만 가지고 있으면 된다. 이를 위해 현재 공개 키를 사용하며, 이 값은 매번 서명 후에 업데이트 된다.

알고리즘 3.9 서명 검증 (다수의 메시지들을 위한 HORSIC)

내부 상태 변수: 현재 공개 키 $(u_1, u_2, \ldots, u_t)$

입력: 메시지 m, 서명 $(c, sig_1, sig_2, \ldots, sig_k)$

출력: "성공" or "실패"

1: $g = G(m)$

2: $(a_1, a_2, \ldots, a_k) = C_{k,z}(g)$

3: $h = H(m|c)$

4: h를 각각의 조각이 $\log_2 t$ 비트가 되도록 k개의 조각으로 분해 $h = h_1 h_2 \ldots h_k$

5: 모든 $j \in \{1, 2, \ldots, k\}$에 대해 h_j를 정수 i_j로 해석

6: 만약 $p \neq q$이고 $i_p = i_q$인 p와 q가 존재하면 $(p, q \in \{1, 2, \ldots, k\})$, "실패"

7: 만약 $f^{a_j}(sig_j) \neq u_{i_j}$인 $j \in \{1, 2, \ldots, k\}$가 존재해도 "실패"

8: 모든 $j \in \{1, 2, \ldots, k\}$에 대해 $u_{i_j} = sig_j$

9: "성공"

3.4 HORSIC의 동작 예

본 절에서는 HORSIC의 동작 과정을 구체적인 예와 함께 설명한다. 보안 매개 변수 값들이 아래의 식 3.4와 같고, 두 개의 메시지 m_1과 m_2가 순서대로 서명되는 경우를 생각해보자.

$$\begin{aligned} l &= 80 \\ t &= 8 \\ w &= 6 \\ k &= 3 \\ z &= 5 \end{aligned} \tag{3.4}$$

송신자는 먼저 8개의 80 비트 난수 $(s_1, s_2, \ldots, s_8)$을 생성한 뒤, 그림 3.2에서와 같이 각각의 s_i에 대해 길이 6의 일방향 체인을 생성한다. 공개 키는 $(f^6(s_1),\ f^6(s_2),\ f^6(s_3),\ f^6(s_4),\ f^6(s_5),\ f^6(s_6),\ f^6(s_7),\ f^6(s_8))$ 이다.

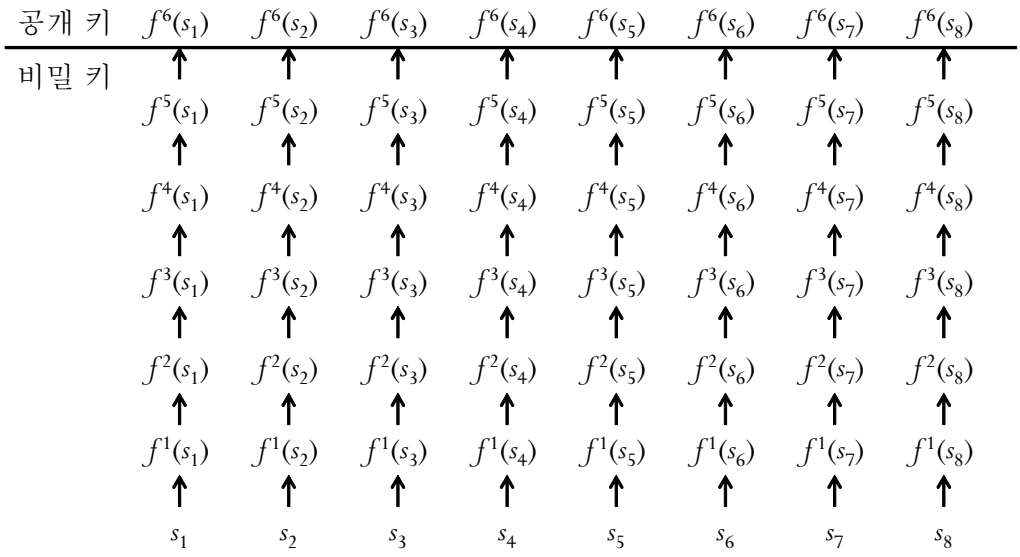

그림 3.2: HORSIC의 동작 예 - 키 생성 후

3.4.1 첫 번째 메시지 m_1에 대한 서명 생성과 검증

첫 번째 메시지 m_1과 관련해 암호학적 해시 함수 H와 G를 적용한 결과가 아래의 식 3.5와 같다고 하자.

$$\begin{aligned} H(m_1 \mid 0) &= 101\ 001\ 110 \\ G(m_1) &= 5 \end{aligned} \tag{3.5}$$

메시지 m_1에 대한 서명을 생성하기 위해 송신자는 우선 위의 식 3.5를 계산한다. 메시지 m_1의 경우 $c = 0$일 때 $i_1 = 5\,(101)$, $i_2 = 1\,(001)$, $i_3 = 6\,(110)$으로 $i_1 \neq i_2 \neq i_3$를 만족시킨다. 기존의 키 경계 지수가 (0, 0, 0, 0, 0, 0, 0, 0)이고 $C_{3,5}(G(m_1)) = C_{3,5}(5) = (3, 1, 1)$이므로 메시지 m_1에 대한 서명은 (0, $f^3(s_5)$, $f^5(s_1)$, $f^5(s_6)$)이 되고, 키 경계 지수는 그림 3.3의 실선이 나타내는 바와 같이 (1, 0, 0, 0, 3, 1, 0, 0)이 된다.

메시지 m_1에 대한 서명 (0, $f^3(s_5)$, $f^5(s_1)$, $f^5(s_6)$)을 검증할 때도 수신자는 우선 위의 식 3.5를 계산한다. 수신자도 역시 $i_1 = 5\,(101)$, $i_2 = 1\,(001)$, $i_3 = 6\,(110)$으로 $i_1 \neq i_2 \neq i_3$를 만족시킴을 알 수 있고, $C_{3,5}(G(m_1)) = C_{3,5}(5) = (3, 1, 1)$인 것도 알 수 있다. 기존의 현재 공개 키가 ($f^6(s_1)$, $f^6(s_2)$, $f^6(s_3)$, $f^6(s_4)$, $f^6(s_5)$, $f^6(s_6)$, $f^6(s_7)$, $f^6(s_8)$)이고 $f^3(f^3(s_5)) = f^6(s_5)$, $f^1(f^5(s_1)) = f^6(s_1)$, $f^1(f^5(s_6)) = f^6(s_6)$이므로 수신자는 받은 서명이 유효함을 알 수 있고, 현재 공개 키는 그림 3.3의 실선이 나타내는 바와 같이 ($f^5(s_1)$, $f^6(s_2)$, $f^6(s_3)$, $f^6(s_4)$, $f^3(s_5)$, $f^5(s_6)$, $f^6(s_7)$, $f^6(s_8)$)이 된다.

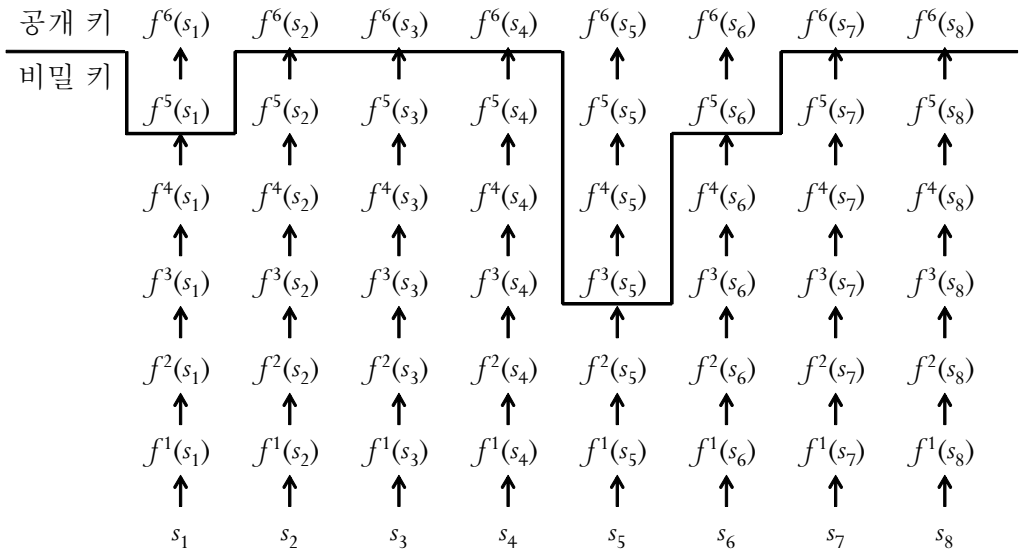

그림 3.3: HORSIC의 동작 예 - 첫 번째 메시지 m_1 서명 후

3.4.2 두 번째 메시지 m_2에 대한 서명 생성과 검증

두 번째 메시지 m_2와 관련해 암호학적 해시 함수 H와 G를 적용한 결과가 아래의 식 3.6과 같다고 하자.

$$\begin{aligned} H(m_2 \mid 0) &= 001\ 111\ 000 \\ G(m_2) &= 3 \end{aligned} \tag{3.6}$$

메시지 m_2에 대한 서명을 생성하기 위해 송신자는 우선 위의 식 3.6을 계산한다. 메시지 m_2의 경우 $c = 0$일 때 $i_1 = 1\ (001)$, $i_2 = 7\ (111)$, $i_3 = 8\ (000)$으로 $i_1 \neq i_2 \neq i_3$를 만족시킨다. 기존의 키 경계 지수가 (1, 0, 0, 0, 3, 1, 0, 0)이고 $C_{3,5}(G(m_2)) = C_{3,5}(3) = (2, 1, 2)$이므로 메시지 m_2에 대한 서명은 (0, $f^3(s_1)$, $f^5(s_7)$, $f^4(s_8)$)이 되고, 키 경계 지수는 그림 3.4의 실선이 나타내는 바와 같이 (3, 0, 0, 0, 3, 1, 1, 2)이 된다.

메시지 m_2에 대한 서명 (0, $f^3(s_1)$, $f^5(s_7)$, $f^4(s_8)$)을 검증할 때도 수신자는 우선 위의 식 3.6을 계산한다. 수신자도 역시 $i_1 = 1\ (001)$, $i_2 = 7\ (111)$, $i_3 = 8\ (000)$으로 $i_1 \neq i_2 \neq i_3$를 만족시킴을 알 수 있고, $C_{3,5}(G(m_2)) = C_{3,5}(3) = (2, 1, 2)$인 것도 알 수 있다. 기존의 현재 공개 키가 ($f^5(s_1)$, $f^6(s_2)$, $f^6(s_3)$, $f^6(s_4)$, $f^3(s_5)$, $f^5(s_6)$, $f^6(s_7)$, $f^6(s_8)$)이고 $f^2(f^3(s_1)) = f^5(s_1)$, $f^1(f^5(s_7)) = f^6(s_7)$, $f^2(f^4(s_8)) = f^6(s_8)$이므로 수신자는 받은 서명이 유효함을 알 수 있고, 현재 공개 키는 그림 3.4의 실선이 나타내는 바와 같이 ($f^3(s_1)$, $f^6(s_2)$, $f^6(s_3)$, $f^6(s_4)$, $f^3(s_5)$, $f^5(s_6)$, $f^5(s_7)$, $f^4(s_8)$)이 된다.

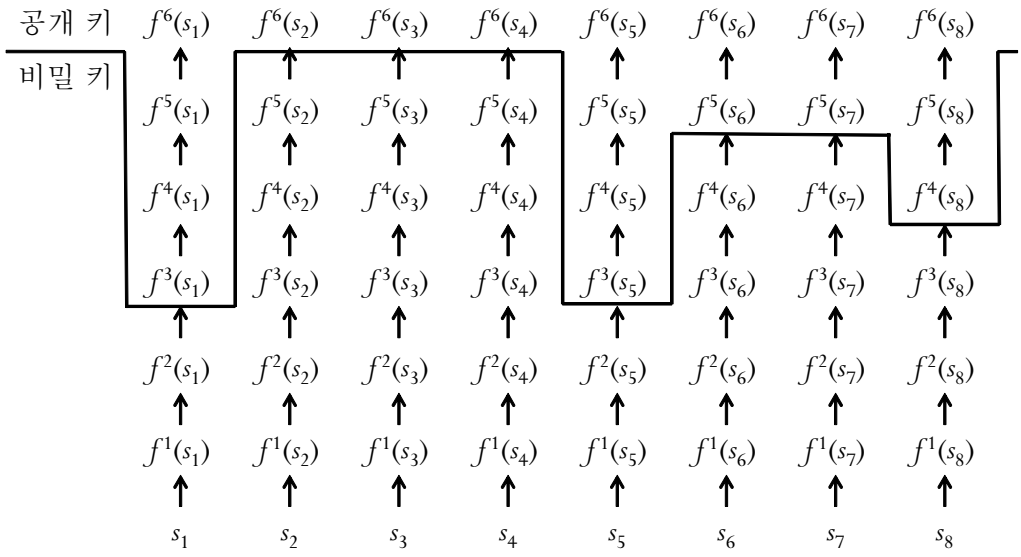

그림 3.4: HORSIC의 동작 예 - 두 번째 메시지 m_2 서명 후

제 4 장 분석 및 비교

본 장에서는 HORSIC을 효율성과 보안성의 관점에서 분석한다. 4.1절에서 HORSIC의 성능을 평가하고, 4.2절에서 HORSIC의 보안성을 분석하며, 마지막으로 4.3절에서 HORSIC의 성능과 보안성을 기존의 다양한 일회용 서명 기법들과 비교한다.

4.1 성능 평가

본 절에서는 HORSIC의 성능을 평가한다. HORSIC의 키 생성, 서명 생성 및 서명 검증 비용은 일방향 함수와 해시 함수의 호출 횟수를 기준으로 계산되었다.

4.1.1 키 생성 비용

HORSIC의 키 생성 과정을 기술한 알고리즘 3.5를 보면 키 생성을 위해 t 개의 난수 s_i를 생성하고 각각의 s_i에 일방향 함수 f를 w 번씩 적용한다. 그러므로 HORSIC은 키 생성 과정에서 일방향 함수 f를 wt 번 호출하며, 따라서 HORSIC의 키 생성 비용은 wt 이다.

4.1.2 서명 생성 비용

다음의 정리 1과 정리 2를 통해 알고리즘 3.8에서 서명 생성을 위해 해시 함수 H가 평균적으로 $\frac{t^k}{t(t-1)...(t-k+1)}$ 번 호출됨을 알 수 있다 [PC06, LC11]. 또한 해시 함수 G도

서명 생성을 위해 한 번 호출되므로, HORSIC의 서명 생성 비용은 $\frac{t^k}{t(t-1)...(t-k+1)} + 1$ 이다.

정리 1 *확률 변수 i_1, i_2, ⋯, i_k 가 모두 같은 확률로 발생 가능한 t 개의 값을 가지는 이산형 균일 분포 (discrete uniform distribution) 를 따른다고 할 때, i_1, i_2, ⋯, i_k 가 모두 서로 다를 확률은 $\frac{t(t-1)...(t-k+1)}{t^k}$ 이다.*

증명 k 개의 확률 변수 각각이 가질 수 있는 경우의 수는 t 이므로 전체 경우의 수는 t^k 이다. 그중 i_1, i_2, ⋯, i_k 가 모두 서로 다른 경우의 수는 $t(t-1)\dots(t-k+1)$ 이다. 그러므로 i_1, i_2, ⋯, i_k 가 모두 서로 다를 확률은 $\frac{t(t-1)...(t-k+1)}{t^k}$ 이다.

정리 2 *알고리즘 3.8에서 서명 생성을 위해 해시 함수 H 는 평균적으로 $\frac{t^k}{t(t-1)...(t-k+1)}$ 번 호출된다.*

증명 $p = \frac{t(t-1)...(t-k+1)}{t^k}$ 이라 하자. 알고리즘 3.8에서 서명 생성을 위해 해시 함수 H 가 한 번만 호출될 확률은 p 이고, 두 번만 호출될 확률은 $(1-p)p$ 이다. 이를 일반화하면 r 번 호출될 확률은 $(1-p)^{r-1}p$ 가 된다. 그러므로 해시 함수 H 는 평균적으로 $\sum_{r=1}^{\infty} r(1-p)^{r-1}p$ 번 호출되고 이 멱급수 (power series) 값을 계산하면 $\frac{1}{p}$ 이다. 따라서 해시 함수 H 는 평균적으로 $\frac{t^k}{t(t-1)...(t-k+1)}$ 번 호출된다.

4.1.3 서명 검증 비용

HORSIC의 서명 검증 과정을 기술한 알고리즘 3.9를 보면 서명 검증을 위해 해시 함수 H 와 G 를 각각 한 번씩 호출하고, 일방향 함수 f 를 $\sum_{i=1}^{k} a_i = z$ 번 호출한다. 따라서 HORSIC의 서명 검증 비용은 $z+2$ 이다.

표 4.1: 비밀 키의 크기와 서명 생성 비용 간의 트레이드 오프

	비밀 키 크기	서명 생성 비용
기본 방법	tlw	$\frac{t^k}{t(t-1)...(t-k+1)} + 1$
비밀 키로 $(s_1, \ldots, s_t)$를 사용	tl	$\frac{t^k}{t(t-1)...(t-k+1)} + k(w-1) + 1$
[CJ03]에서 제안한 방법	$tl \log w$	$\frac{t^k}{t(t-1)...(t-k+1)} + k \log(w-1) + 1$

4.1.4 공개 키, 비밀 키 및 서명의 크기

HORSIC의 키 생성 과정을 기술한 알고리즘 3.5를 보면 공개 키의 크기는 tl 비트이고, 비밀 키의 크기는 tlw 비트이다. 이와 같이 HORSIC은 크기가 tlw 비트인 비밀 키를 가지는데, 만약 이 크기가 실제 센서 네트워크 환경에서 사용하기에 너무 크다고 생각되면 비밀 키를 $(s_1, s_2, \ldots, s_t)$로 하여 tl 비트로 줄일 수도 있다. 대신 비밀 키를 $(s_1, s_2, \ldots, s_t)$로 하면 서명 생성 비용의 상계 (upper bound)가 $\frac{t^k}{t(t-1)...(t-k+1)} + k(w-1) + 1$로 늘어난다. 비밀 키의 크기와 서명 생성 비용 둘 다 적절히 줄일 수 있는 방법도 있다 [CJ03]. 이 방법을 사용하면 비밀 키의 크기는 $tl \log w$ 비트가 되고, 서명 생성 비용은 $\frac{t^k}{t(t-1)...(t-k+1)} + k \log(w-1) + 1$가 된다. 표 4.1은 이러한 비밀 키의 크기와 서명 생성 비용 간의 트레이드 오프를 나타낸다.

알고리즘 3.8에서 c의 값은 (해시 함수 H의 호출 횟수 - 1)이므로 c의 평균 값은 $\frac{t^k}{t(t-1)...(t-k+1)} - 1$이다. 그러므로 서명의 크기는 $kl + |c| = kl + \log_2(\frac{t^k}{t(t-1)...(t-k+1)} - 1)$ 비트인데, $\log_2(\frac{t^k}{t(t-1)...(t-k+1)} - 1)$ 값이 kl에 비해 굉장히 작으므로 서명의 크기는 대략 kl 비트라고 할 수 있다.

4.1.5 서명 가능한 메시지의 수

키 생성 과정에서 전체 wt 개의 일방향 체인 값을 생성하고, 각각의 서명이 z 개의 일방향 체인 값을 드러내므로 HORSIC이 서명할 수 있는 메시지 수의 상계 (upper bound)는 $\frac{wt}{z}$ 이다. 하지만 한 위치에 w 개의 일방향 체인 값이 있고, 한 번의 서명을 통해 한 위치에서 최대 $z-k+1$ 개의 일방향 체인 값이 드러날 수 있으므로, 최악의 경우 $\frac{w}{z-k+1}$ 개의 메시지만 서명 가능할 수도 있다. 따라서 HORSIC이 서명할 수 있는 메시지의 정확한 개수는 서명이 이루어지면서 키 경계가 어떻게 변화하느냐에 따라 최소 $\frac{w}{z-k+1}$ 개에서 최대 $\frac{wt}{z}$ 개 사이로 결정된다.

그림 4.1은 w와 t가 고정되어 있고 k가 6, 7, 8, 9, 10일 때 z의 변화에 따른 서명 가능한 메시지 수의 변화를 나타낸다. 그림 4.1의 그래프에서 가로 축은 z 값을 나타내고, 세로 축은 서명 가능한 메시지의 수를 나타내며, 공개 키의 크기와 관련된 보안 매개 변수 t는 1024로, 키 생성 비용 및 서명 가능한 메시지의 수와 관련된 보안 매개 변수 w는 1000으로 고정되었다. 또한 서명 가능한 메시지의 수를 정확히 알아보기 위해 각각의 k와 z 값에 대해 실험을 100번씩 수행하고 그 평균을 그래프에 나타내었다.

그림 4.1의 그래프에서 z 값이 커짐에 따라 서명 가능한 메시지의 수가 줄어드는 것을 확인할 수 있다. z 값이 커지면 매번 서명할 때마다 드러나는 일방향 체인 값의 수가 많아지기 때문에 다른 보안 매개 변수 값들이 일정하다면 서명 가능한 메시지의 수는 줄어들 수 밖에 없다. 또한 같은 z 값에 대해 약간의 차이이긴 하지만 k 값이 클 수록 서명 가능한 메시지의 수가 늘어나는 것도 확인할 수 있다. k 값이 커질수록 여러 위치에 분산되어 있는 일방향 체인 값들이 고르게 드러나기 때문이다. HORSIC에서 실제로 서명 가능한 메시지의 수는 z 값이 15일 때는 가능한 최대 값인 $\frac{wt}{z}$의 약

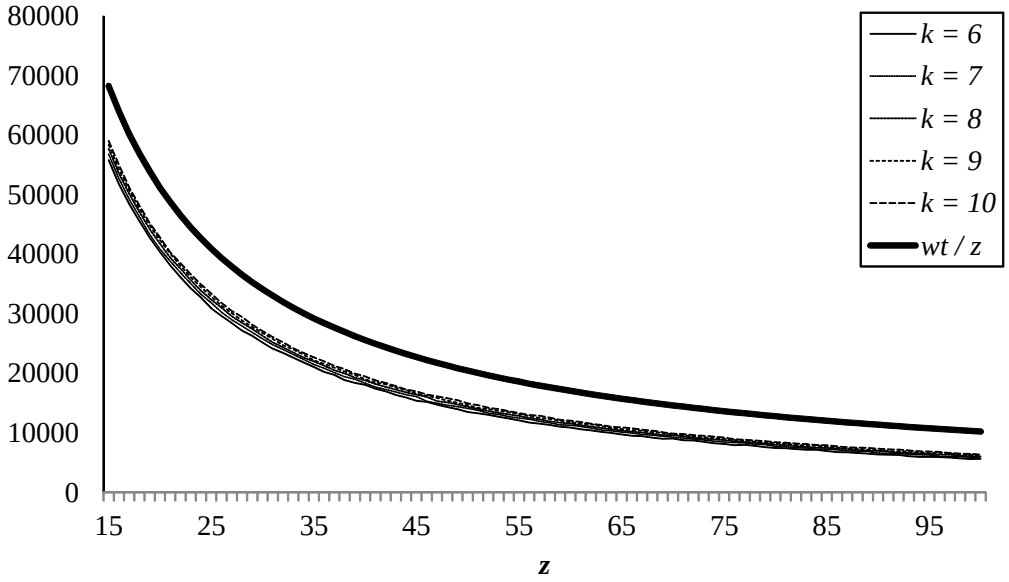

그림 4.1: z의 변화에 따른 서명 가능한 메시지 수의 변화

86 % 정도이지만, z 값이 증가함에 따라 이 비율도 감소하면서 z 값이 100일 때는 약 62 % 정도가 된다.

그림 4.2는 t와 z가 고정되어 있고 k가 6, 7, 8, 9, 10일 때 w의 변화에 따른 서명 가능한 메시지 수의 변화를 나타낸다. 그림 4.2의 그래프에서 가로 축은 w 값을 나타내고, 세로 축은 서명 가능한 메시지의 수를 나타내며, 공개 키의 크기와 관련된 보안 매개 변수 t는 1024로, 서명 위조 확률 및 서명 가능한 메시지의 수와 관련된 보안 매개 변수 z는 50으로 고정되었다. 또한 서명 가능한 메시지의 수를 정확히 알아보기 위해 각각의 k와 w 값에 대해 실험을 100번씩 수행하고 그 평균을 그래프에 나타내었다.

그림 4.2의 그래프에서 w 값이 커짐에 따라 서명 가능한 메시지의 수도 많아지는 것을 확인할 수 있다. 키 생성 과정에서 전체 wt개의 일방향 체인 값을 생성하기

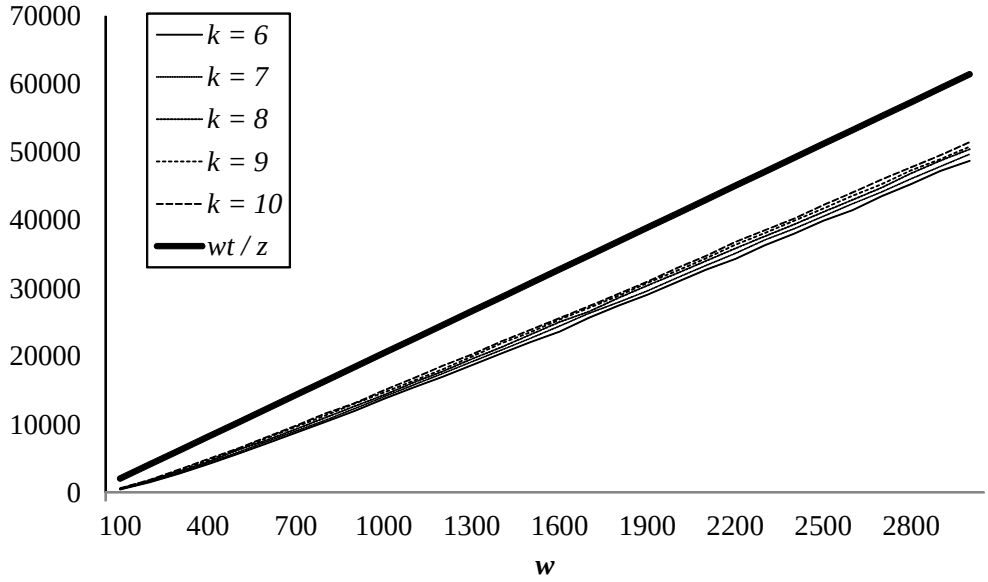

그림 4.2: w의 변화에 따른 서명 가능한 메시지 수의 변화

때문에 w 값이 커지면 서명 생성을 위해 사용할 수 있는 일방향 체인 값이 많아져 서명 가능한 메시지의 수도 증가하기 때문이다. 또한 앞의 그림 4.1의 그래프에서와 마찬가지로 같은 w 값에 대해 약간의 차이이긴 하지만 k 값이 클 수록 서명 가능한 메시지의 수가 늘어나는 것도 확인할 수 있다. HORSIC에서 실제로 서명 가능한 메시지의 수는 w 값이 100일 때는 가능한 최대 값인 $\frac{wt}{z}$의 약 21.5 ~ 31.5 % 정도이지만, w 값이 증가함에 따라 이 비율도 증가하면서 w 값이 3000일 때는 약 79 ~ 84 % 정도가 된다.

서명 가능한 메시지의 수는 실험이 아닌 수식 분석을 통해서도 구할 수 있다. HORSIC의 서명 생성 과정을 표현한 그림 3.1에서 볼 수 있다시피, HORSIC은 $H(m|c)$를 통해 t개의 s_i들 중 k개의 s_i들을 메시지에 따라 서로 다르게 골고루 선택한다. 따라서 한 번 서명이 이루어질 때마다 각각의 s_i가 선택될 확률은 $\frac{k}{t}$가 되고, 서명이 n번 이루어졌을 때 각각의 s_i가 선택된 횟수를 나타내는 확률 분포는 이항 분포 (binomial

distribution) $B(n, \frac{k}{t})$를 따른다.

확률 변수 (random variable) G가 구간 $[0, \binom{z-1}{k-1})$에서 균등 분포 (uniform distribution)를 따른다고 하자. 이 때 식 $z = \sum_{i=1}^{k} a_i$ (a_i는 1보다 크거나 같은 정수)의 G 번째 해를 구성하는 각 a_i 값들을 확률 변수 X로 나타내면, X의 정확한 확률 분포는 식 3.3에 의해 다음과 같이 구해진다.

$$
\begin{aligned}
Pr(X=1) &= \frac{\binom{z-2}{k-2}}{\binom{z-1}{k-1}} = \frac{k-1}{z-1} \\
Pr(X=2) &= \frac{\binom{z-3}{k-2}}{\binom{z-1}{k-1}} = \frac{k-1}{z-1} \times \frac{z-k}{z-2} \\
Pr(X=3) &= \frac{\binom{z-4}{k-2}}{\binom{z-1}{k-1}} = \frac{k-1}{z-1} \times \frac{z-k}{z-2} \times \frac{z-k-1}{z-3} \\
Pr(X=4) &= \frac{\binom{z-5}{k-2}}{\binom{z-1}{k-1}} = \frac{k-1}{z-1} \times \frac{z-k}{z-2} \times \frac{z-k-1}{z-3} \times \frac{z-k-2}{z-4} \\
&\ldots \\
Pr(X=z-k+1) &= \frac{\binom{k-2}{k-2}}{\binom{z-1}{k-1}} = \frac{k-1}{z-1} \times \frac{z-k}{z-2} \times \frac{z-k-1}{z-3} \times \ldots \times \frac{1}{k-1}
\end{aligned}
\tag{4.1}
$$

하지만 이러한 확률 분포는 너무 복잡해 이것을 가지고는 더 이상 분석이 불가능하다. 따라서 계산을 간단히 하기 위해 식 4.1을 이항 분포로 근사하여 이를 $B(z, \frac{1}{k})$로 나타내었다. 이는 k개의 상자 중 하나를 매번 임의로 선택하여 공을 넣는 일을 z번 수행하였을 때 각각의 상자에 들어간 공의 개수를 나타내며 식 4.1과 비슷한 분포를 가진다.

앞에서 이야기한 바와 같이 서명이 n번 이루어졌을 때 각각의 s_i가 선택된 횟수를 나타내는 확률 분포는 이항 분포 $B(n, \frac{k}{t})$를 따르고, 각각의 s_i가 선택되었을 때 공개되는 키의 개수를 나타내는 확률 분포는 이항 분포 $B(z, \frac{1}{k})$를 따른다. 따라서

서명이 n 번 이루어졌을 때 각각의 s_i 가 공개하는 키의 개수를 나타내는 확률 분포는 $B(z \times B(n, \frac{k}{t}), \frac{1}{k}) \approx B(nz, \frac{1}{t})$ 이 된다. 이 확률 분포의 평균은 $\frac{nz}{t}$ 이고, 분산은 $\frac{nz(t-1)}{t^2}$ 이므로, 이항 분포 $B(nz, \frac{1}{t})$ 은 정규 분포 $N(\frac{nz}{t}, \frac{nz(t-1)}{t^2})$ 로 근사할 수 있다.

HORSIC에서 서명 가능한 메시지의 수는 서로 독립인 t 개의 확률 변수들이 모두 정규 분포 $N(\frac{nz}{t}, \frac{nz(t-1)}{t^2})$ 을 따를 때, 이 확률 변수들이 모두 w 를 넘지 않을 확률이 0.5보다 크도록 하는 가장 큰 n 값이다. 여기서 같은 분포를 가지는 t 개의 확률 변수들 중 최대 값을 나타내는 확률 변수의 누적 분포 함수(cumulative distribution function) $F_{max}(y)$ 는 다음과 같이 계산된다.

$$F_{max}(y) = F_X(y)^t \tag{4.2}$$

확률 변수들의 최대 값의 평균이 w 를 넘지 않아야 하므로 $F_{max}(y) = 0.5$ 라 하고, 앞의 실험 결과와 비교하기 위해 $t = 1024$ 라 하면 $F_X(y)$ 는 다음과 같이 계산된다.

$$\begin{aligned} 0.5 &= F_X(y)^{1024} \\ F_X(y) &= 0.999323 \end{aligned} \tag{4.3}$$

정규 분포표에서 누적 분포 함수 값이 0.999323에 해당하는 표준 편차 값을 찾으면 약 3.2이다. 따라서 $t = 1024$ 일 때 다음과 같은 식이 성립한다.

$$w = \frac{nz}{t} + 3.2 \times \frac{\sqrt{nz(t-1)}}{t} \tag{4.4}$$

여기에 $t = 1024$을 대입하고 식을 정리하면 다음과 같다.

$$nz + 102.35\sqrt{nz} - 1024w = 0 \tag{4.5}$$

마지막으로 위의 식을 $\sqrt{nz}$에 대한 2차 방정식으로 계산하여 n 값을 구한다.

$$\begin{aligned} \sqrt{nz} &= \frac{-102.35 + \sqrt{102.35^2 + 4 \times 1024w}}{2} \\ &= \frac{-102.35 + \sqrt{4096w + 10475.5}}{2} \end{aligned} \tag{4.6}$$

$$\begin{aligned} nz &= \frac{4096w + 10475.5 + 10475.5 - 204.7\sqrt{4096w + 10475.5}}{4} \\ &= 1024w + 5237.75 - 51.175\sqrt{4096w + 10475.5} \end{aligned} \tag{4.7}$$

$$n = \frac{1024w + 5237.75 - 51.175\sqrt{4096w + 10475.5}}{z} \tag{4.8}$$

그림 4.3과 그림 4.4의 그래프는 식 4.8을 통해 계산된 HORSIC에서 서명 가능한 메시지의 수를 각각 그림 4.1과 그림 4.2의 그래프와 비교한 것이다. 식 4.8을 통해 계산된 결과와 실험을 통해 구해진 결과가 꽤 많은 오차를 보이는 것을 알 수 있는데, 그 이유들을 분석해보면 다음과 같다.

- 계산을 간단히 하기 위해 식 4.1의 확률 분포를 이항 분포 $B(z, \frac{1}{k})$로 근사
- 이항 분포 $B(z \times B(n, \frac{k}{t}), \frac{1}{k})$을 이항 분포 $B(nz, \frac{1}{t})$로 근사

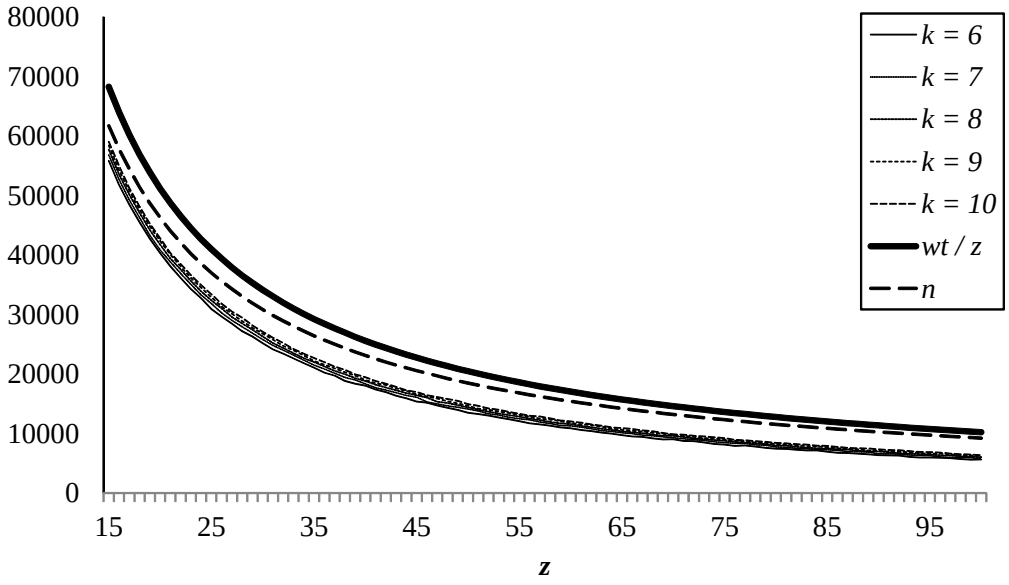

그림 4.3: z의 변화에 따른 서명 가능한 메시지 수의 변화 - 수식과 비교

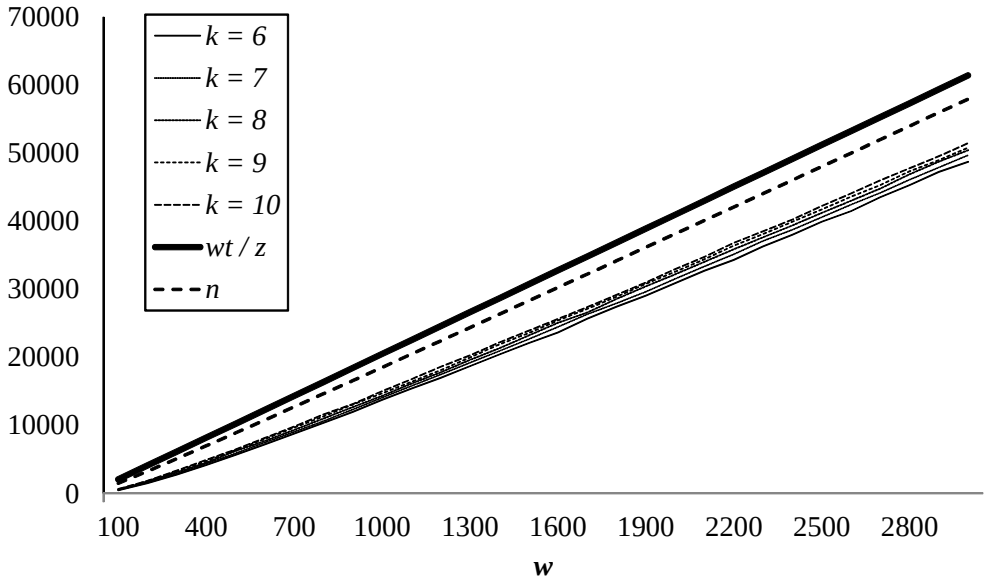

그림 4.4: w의 변화에 따른 서명 가능한 메시지 수의 변화 - 수식과 비교

- 이항 분포 $B(nz, \frac{1}{t})$를 정규 분포 $N(\frac{nz}{t}, \frac{nz(t-1)}{t^2})$로 근사

4.2 보안성 분석

본 절에서는 공격자가 비적응적 메시지 공격 (non-adaptive message attack)을 수행할 때 발생하는 서명 위조 확률을 계산함으로써 HORSIC의 보안성을 분석한다. 비적응적 메시지 공격에서는 공격자가 해시 함수 G, H와는 무관하게 메시지를 선택하여 그에 대한 서명을 얻고, 이를 기반으로 새로운 메시지에 대한 서명 생성을 시도한다. 공격자가 수행할 수 있는 더 강력한 공격인 적응적 선택 메시지 공격 (adaptive chosen message attack)은 기존 논문들 ([Per01, MP02, RR02, PC06, LC11])에서와 마찬가지로 본 논문에서도 고려하지 않았다.

공격자가 유효한 서명 $(c, sig_1, sig_2, \ldots, sig_k)$를 가지고 있다고 가정하자. HORSIC에서 모든 유효한 서명은 서명 전 이미 드러난 일방향 체인 값 (현재 공개 키)에서 z개의 새로운 일방향 체인 값을 추가로 드러낸다. 여기서 공격자가 $j \in \{1, 2, \ldots, k\}$인 어떠한 j에 대해 sig_j 대신 $f^i(sig_j)$를 새로운 서명을 생성하는데 사용하고자 한다고 하자 (여기서 i는 1보다 크거나 같은 정수이다). 이 경우 $f^i(sig_j)$를 통해 드러나는 일방향 체인 값은 sig_j일 때에 비해 i개 만큼 줄어들며, 따라서 다른 부분에서 일방향 체인 값을 더 드러내야 한다. 이를 위해서는 f의 역함수를 계산해야 하는데 이는 일방향 함수의 정의에 의해 불가능하며, 따라서 공격자는 집합 $\{sig_1, sig_2, \ldots, sig_k\}$의 요소들만 사용해서 서명을 위조해야 한다.

공격자는 집합 $\{sig_1, sig_2, \ldots, sig_k\}$의 요소들을 서로 다른 순서로 배치해 $k!$개의 서명을 만들 수 있다. 공격자가 $H(m|c) = h_1h_2h_3\ldots h_k$이고 $C_{k,z}(G(m)) = (a_1, a_2, a_3, \ldots, a_k)$인 메시지 m에 대한 유효한 서명 $(c, sig_1, sig_2, sig_3, \ldots, sig_k)$를 가지고 있다고 가정하자. 만약 $H(m'|c') = h_2h_1h_3\ldots h_k$이고 $C_{k,z}(G(m')) = (a_2, a_1, a_3, \ldots, a_k)$인 메시지 m'이 존재하면, $(c', sig_2, sig_1, sig_3, \ldots, sig_k)$는 메

시지 m'에 대한 유효한 서명이 된다.

정해진 값 h에 대해 $H(m'|c') = h$일 확률은 $\frac{1}{2^{|H()|}} = \frac{1}{2^{k \log_2 t}} = \frac{1}{t^k}$이고, 정해진 값 g에 대해 $G(m') = g$일 확률은 $\frac{1}{\binom{z-1}{k-1}} = \frac{(k-1)!(z-k)!}{(z-1)!}$이다. 따라서 HORSIC에서의 서명 위조 확률은 $k! \times \frac{1}{t^k} \times \frac{(k-1)!(z-k)!}{(z-1)!} = \frac{k!(k-1)!(z-k)!}{t^k(z-1)!}$가 된다.

4.3 다른 기법들과의 비교

본 절에서는 HORSIC의 성능과 보안성을 기존의 다양한 일회용 서명 기법들과 비교한다. 각 기법들의 성능 평가를 위해 키 생성 비용, 서명 생성 비용, 서명 검증 비용, 서명 크기 및 공개 키 크기를 조사하였고, 보안성 분석을 위해 공격자의 위조 확률을 계산하였다.

표 4.2는 HORSIC의 성능과 보안성을 같은 공개 키 크기 ($= tl$)와 거의 같은 서명 크기 ($\approx kl$)를 가진 다른 다양한 일회용 서명 기법들 (BiBa [Per01], Powerball [MP02], HORS [RR02], Park [PC06], TSV (HSLV와 LSHV의 특별한 경우 포함) [LC11])과 비교한 결과이다. 여기서 사용된 다양한 보안 매개 변수들과 기호들의 의미는 표 4.3과 표 4.4에 각각 나타내었다.

표 4.2에서 보다시피 같은 공개 키 크기와 거의 같은 서명 크기를 가질 때 HORSIC의 키 생성 비용은 wt로 다른 일회용 서명 기법들에 비해 큰 편이다. 하지만 키 생성 과정은 한 번만 수행되며, 오프라인으로도 가능하고, 쉽게 병렬화될 수도 있다. 그러므로 HORSIC의 키 생성 과정이 그다지 큰 부담은 되지 않는다.

HORSIC을 제외한 다른 일회용 서명 기법들의 서명 위조 확률은 최소 $\frac{1}{t^k}$이고, HORSIC의 서명 위조 확률은 $\frac{1}{t^k} \times k! \,/\, \binom{z-1}{k-1} = \frac{k!(k-1)!(z-k)!}{t^k(z-1)!}$이다. 따라서 보안 매개 변수 z가 $\binom{z-1}{k-1} \gg k!$이 되도록 충분히 크게 설정되면 HORSIC의 서명 위조 확률은 다른 일회용 서명 기법들에 비해 크게 작아진다.

그림 4.5은 k가 6, 7, 8, 9, 10일 때 z의 변화에 따른 $\frac{k!(k-1)!(z-k)!}{(z-1)!}$ 값의 변화를 나타낸다. 그림 4.5의 그래프에서 가로 축은 z 값을 나타내고, 세로 축은 로그 스케일 (logarithmic scale)로 $\frac{k!(k-1)!(z-k)!}{(z-1)!}$ 값을 나타낸다. 그래프에서 z 값이 커짐에 따라

표 4.2: 같은 공개 키 크기와 거의 같은 서명 크기를 가진 다양한 일회용 서명 기법들의 비교

서명 기법	키 생성 비용	서명 생성 비용	서명 검증 비용	서명 크기	공개 키 크기	위조 확률
BiBa	t	$2t$	$2k+1$	kl	tl	$\frac{k!}{2t^k}$
Powerball	$2t$	$2t$	$2k+1$	kl	tl	$\frac{(k-1)!}{2t^k}$
HORS	t	1	$k+1$	kl	tl	$\frac{k^k}{t^k}$
Park	$2t$	ρ	$\frac{3}{2}k+1$	$kl+\log_2\rho$	tl	$\frac{1}{t^k}$
HSLV	t	$k!\mu$	$k+1$	$kl+\log_2(k!\mu)$	tl	$\frac{1}{t^k}$
LSHV	kt	μ	$\frac{k(k+1)}{2}+1$	$kl+\log_2\mu$	tl	$\frac{1}{t^k}$
TSV	$(\max\{w_1,\ldots,w_g\}+1)t$	τ	$1+k+\sum_{r=1}^{g} n_r w_r$	$kl+\log_2\tau$	tl	$\frac{1}{t^k}$
HORSIC	wt	$\mu+1$	$z+2$	$kl+\log_2\mu$	tl	$\frac{k!(k-1)!(z-k)!}{t^k(z-1)!}$

표 4.3: (표 4.2에서 사용된) 다양한 보안 매개 변수들의 의미

보안 매개 변수	의미
l	일방향 함수 f의 보안성을 결정
t	보안 매개 변수 l과 함께 공개 키의 크기를 결정
k	보안 매개 변수 l과 함께 서명의 크기를 결정
g	TSV에 고유한 보안 매개 변수
w_r	TSV에 고유한 보안 매개 변수
n_r	TSV에 고유한 보안 매개 변수
w	HORSIC에서 키 생성 비용과 서명 가능한 메시지의 수를 결정
z	HORSIC에서 서명 위조 확률과 서명 가능한 메시지의 수를 결정

표 4.4: (표 4.2에서 사용된) 다양한 기호들의 의미

기호	의미
ρ	$\frac{t^k(\frac{k}{2}!)^2(t-k)!}{t!}$
μ	$\frac{t^k}{t(t-1)...(t-k+1)}$
τ	$\mu\prod_{r=1}^{g}(n_r!)$

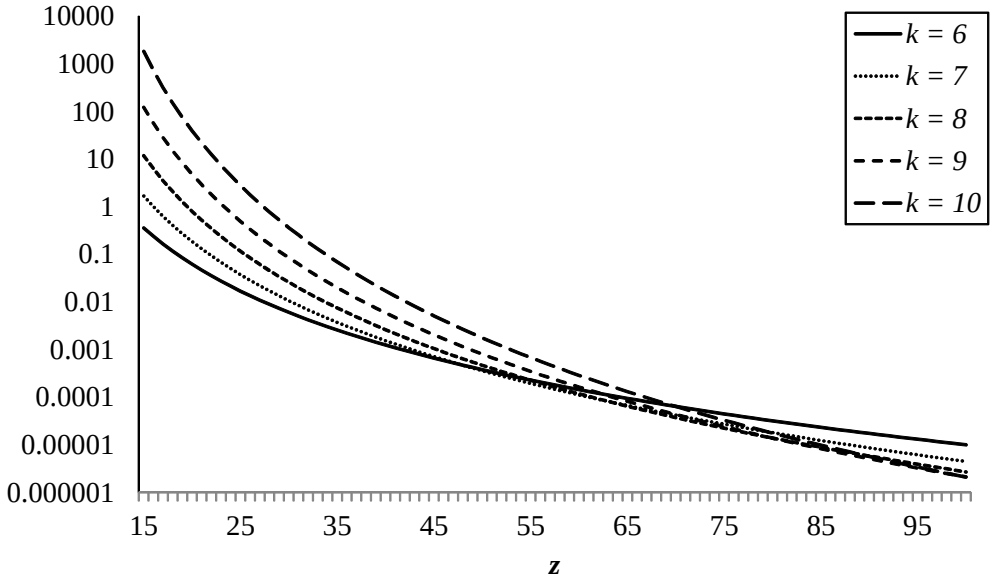

그림 4.5: z의 변화에 따른 $\frac{k!(k-1)!(z-k)!}{(z-1)!}$ 값의 변화

$\frac{k!(k-1)!(z-k)!}{(z-1)!}$ 값이 기하급수적으로 줄어드는 것을 확인할 수 있으며, 그 값이 줄어드는 정도는 k 값이 클 수록 더 급격해진다. $\frac{k!(k-1)!(z-k)!}{(z-1)!}$ 값은 k가 6, 7, 8, 9, 10일 때 z가 각각 13, 17, 20, 24, 28이 넘으면서부터 1보다 작아지기 시작한다. 그래프에서 보다시피 HORSIC의 서명 위조 확률은 z 값이 커질수록 다른 일회용 서명 기법들에 비해 크게 작아진다. 하지만 z 값이 커질수록 서명 검증 비용도 증가하며 그림 4.1에서 보다시피 서명 가능한 메시지의 수도 줄어든다.

표 4.5는 다양한 보안 매개 변수의 HORSIC을 같은 보안 수준 (80 비트)을 가진 다양한 일회용 서명 기법들과 비교한 것이다. 일반적으로 보안 수준은 80 비트이면 충분하다고 알려져 있다 [MP02]. 표 4.5의 모든 기법에서 보안 매개 변수 l은 80이다. BiBa, Powerball, HORS, Park에서 보안 매개 변수 t는 1024로 설정되었으며, 보안 매개 변수 k는 서명 위조 확률을 $\frac{1}{2^{80}}$보다 작도록 제한하면서 가능한 한 최소화했다.

TSV의 경우 보안 매개 변수 t는 128로, 보안 매개 변수 k는 13으로 설정되었다. HORSIC(t, k, z, w)는 보안 매개 변수를 각각 t, k, z, w로 설정한 HORSIC을 나타낸다.

표 4.5에 있는 일회용 서명 기법들의 서명 위조 확률은 HORSIC을 제외하고 모두 $\frac{1}{t^k}$보다 크거나 같다. 이 경우 서명 위조 확률이 $\frac{1}{2^{80}}$ 이하가 되기 위해서는 $t^k \geq 2^{80}$ 이어야 하고, 양변에 $\log_2$를 적용하면 $k \log_2 t \geq 80$이 된다. 여기서 t는 공개 키의 크기를 결정하고, k는 서명의 크기를 결정하므로 앞의 식에 의하면 공개 키의 크기와 서명의 크기를 동시에 줄이는 것은 불가능하다. 즉 HORSIC을 제외한 다른 일회용 서명 기법들에서, 공개 키의 크기를 1024×80 비트로 고정시킨 상태에서 서명의 크기를 8×80 비트 미만으로 줄이거나, 서명의 크기를 8×80 비트로 고정시킨 상태에서 공개 키의 크기를 1024×80 비트 미만으로 줄이는 것은 불가능하다.

하지만 센서 네트워크 환경에서 공개 키의 크기와 서명의 크기는 둘 다 가능한 한 작을 수록 좋다. 매우 제한적인 저장 공간을 가진 센서 노드에게 있어 1024×80 비트 ($= 10$ KB)의 공개 키 크기는 꽤 큰 편이다 [HBH05]. 또한 센서 노드는 매우 제한적인 에너지를 가지는데 특히 무선 통신을 위해 많은 에너지를 소비한다. 그러므로 서명의 크기 또한 가능한 한 작을 수록 좋다.

HORSIC은 공개 키의 크기를 그대로 유지한 상태에서 키 생성 비용과 서명 검증 비용을 늘리는 대신 서명의 크기를 줄일 수 있다. 마찬가지로 서명의 크기를 그대로 유지한 상태에서 키 생성 비용과 서명 검증 비용을 늘리는 대신 공개 키의 크기를 줄일 수도 있다. 물론 키 생성 비용과 서명 검증 비용을 늘리는 대신 공개 키의 크기와 서명의 크기를 어느 정도 동시에 줄이는 것도 가능하다.

비록 HORSIC의 서명 검증 비용이 다른 일회용 서명 기법들에 비해 크긴 하지만,

표 4.5: 같은 보안 수준 (80 비트)을 가진 다양한 일회용 서명 기법들의 비교

서명 기법	키 생성 비용	서명 생성 비용	서명 검증 비용	서명 크기	공개 키 크기	위조 확률
BiBa	1024	2048	23	11×80	1024×80	$\leq \frac{1}{2^{80}}$
Powerball	2048	2048	21	10×80	1024×80	$\leq \frac{1}{2^{80}}$
HORS	1024	1	14	13×80	1024×80	$\leq \frac{1}{2^{80}}$
Park	2048	592	13	8×80	1024×80	$\leq \frac{1}{2^{80}}$
TSV	512	1.2×10^5	24	13×80	128×80	$\leq \frac{1}{2^{80}}$
HORSIC (1024, 8, 20, 13)	13312	≈ 2	22	8×80	1024×80	$\leq \frac{1}{2^{80}}$
HORSIC (1024, 7, 43, 37)	37888	≈ 2	45	7×80	1024×80	$\leq \frac{1}{2^{80}}$
HORSIC (512, 9, 22, 14)	7168	≈ 2	24	9×80	512×80	$\leq \frac{1}{2^{80}}$
HORSIC (512, 8, 39, 32)	16384	≈ 2	41	8×80	512×80	$\leq \frac{1}{2^{80}}$
HORSIC (256, 10, 28, 19)	4864	≈ 2	30	10×80	256×80	$\leq \frac{1}{2^{80}}$
HORSIC (256, 9, 42, 34)	8704	≈ 2	44	9×80	256×80	$\leq \frac{1}{2^{80}}$
HORSIC (128, 11, 38, 28)	3584	≈ 2	40	11×80	128×80	$\leq \frac{1}{2^{80}}$
HORSIC (128, 10, 54, 45)	5760	≈ 2	56	10×80	128×80	$\leq \frac{1}{2^{80}}$

표 4.6: Mica2Dot 센서 플랫폼에서의 에너지 비용

	에너지 비용	
데이터 전송	59.2 μJ/byte (송신)	28.6 μJ/byte (수신)
AES-128	1.62 μJ/byte (암호화)	2.49 μJ/byte (복호화)
RSA-1024	304 mJ (서명)	11.9 mJ (검증)
RSA-2048	2302.7 mJ (서명)	53.7 mJ (검증)
ECDSA-160	22.82 mJ (서명)	45.09 mJ (검증)
ECDSA-224	61.54 mJ (서명)	121.98 mJ (검증)

센서 노드에서 전체적으로 소모되는 에너지의 양은 HORSIC의 경우 서명의 크기가 줄어듬으로 인해 다른 일회용 서명 기법들에 비해 오히려 더 적은 경우가 많다.

표 4.6은 [WGE+05]에서 Crossbow 사의 Mica2Dot 센서 플랫폼을 대상으로 측정한 각 연산의 에너지 비용을 나타낸다. 일방향 함수 f의 경우 한 번의 AES-128 암호화 연산으로 구현될 수 있으므로 [PGV94], 일방향 함수 f를 한 번 호출하는데 필요한 에너지의 양은 1.62 μJ/byte $\times$ 128 bit = 1.62 μJ/byte $\times$ 16 byte = 25.92 μJ 정도로 추정된다.

무선 센서 네트워크에서 브로드캐스트 통신의 경우 대부분의 노드들이 중간 노드의 역할을 하면서 패킷을 받으면 이를 다른 노드들에게 전달한다. 그러므로 대부분의 노드들은 송신과 수신을 둘 다 수행한다. 따라서 서명의 크기를 80 비트 줄임으로써 절약하는 에너지의 양은 (59.2 μJ/byte + 28.6 μJ/byte) $\times$ 80 bit = 87.8 μJ/byte $\times$ 10 byte = 878 μJ 정도로 추정되며, 이는 일방향 함수 f를 호출하는데 필요한 에너지의

양인 25.92 μJ의 대략 34배가 된다.

표 4.6을 보면 RSA의 경우 서명보다 검증 시에 훨씬 더 적은 에너지를 사용하는 것을 알 수 있는데, 이는 공개 키를 계산하기 간단한 값으로 정할 수 있기 때문이다 [Koc94]. 또한 ECDSA의 경우 서명보다 검증 시에 대략 두 배의 에너지를 필요로 하는 것을 알 수 있는데, 이는 필요한 스칼라 곱셈 연산의 횟수 차이 때문이다. RSA와 ECDSA를 통틀어서 서명 검증 시 가장 적은 에너지를 사용하는 경우는 RSA-1024의 경우로 11.9 mJ을 사용하고, 이는 일방향 함수 f를 호출하는데 필요한 에너지의 양인 25.92 μJ의 대략 459배이다. 가장 많은 에너지를 사용하는 경우는 ECDSA-224의 경우로 121.98 mJ을 사용하고, 이는 일방향 함수 f를 호출하는데 필요한 에너지의 대략 4706배이다. 그러므로 RSA나 ECDSA 같은 공개 키 기반 전자 서명 기법들은 일회용 서명 기법들에 비해 굉장히 많은 양의 에너지를 필요로 함을 알 수 있다.

제 5 장 결론

무선 센서 네트워크는 보통 수백에서 수천 개의 센서 노드들로 구성되기 때문에 각 센서 노드는 가능한 한 최소한의 비용으로 제작되어야 하며, 따라서 매우 제한된 컴퓨팅 파워와 메모리, 에너지를 가진다. 브로드캐스트 인증은 이러한 무선 센서 네트워크에서 가장 기본적이면서도 중요한 보안 요소이지만, 센서 노드의 자원 제약적인 특성으로 인해 기존의 인증 기법들은 대부분 사용되지 못한다. 일대일 통신에서 사용했던 MAC을 통한 방법은 같은 키를 가진 수신자가 메시지를 위조할 수 있기 때문에 부적절하고, RSA나 ECDSA 같은 공개 키 기반 서명 기법은 너무 많은 계산량을 요구한다. μTESLA는 일방향 키 체인과 지연 키 공개 기법을 통해 오직 대칭적인 요소들만을 사용해서 효율적인 브로드캐스트 인증을 수행하지만, 노드들 간의 시간 동기화를 필요로 할 뿐만 아니라 인증 시간의 지연도 발생한다.

무선 센서 네트워크에서 브로드캐스트 인증은 일회용 서명 기법을 통해서도 가능하다. 일회용 서명 기법은 일방향 함수만을 기반으로 하므로 공개 키 기반 서명 기법들보다 서명 생성과 검증에 많은 계산을 필요로 하지 않아 무선 센서 네트워크 환경에 적합하다. 더구나 기존의 일반적인 공개 키 기반 서명 기법들은 양자 컴퓨터가 본격적으로 실용화 된다면 더 이상 사용할 수 없게 되지만, 일방향 함수를 기반으로 하는 일회용 서명 기법들은 여전히 사용 가능하다. 하지만 이름 그대로 주어진 비밀 키로 오직 한 번만 서명이 가능하고, 공개 키의 저장을 위해 많은 양의 메모리를 필요로 하며 서명의 크기가 커서 높은 통신 부하를 야기한다.

본 논문에서는 HORSIC이라 부르는 무선 센서 네트워크에서 브로드캐스트 인증을 위한 효율적인 일회용 서명 기법을 제안하였다. HORSIC은 기존에 제안된 HORS의 확장이다. HORS는 t개의 원소를 가지는 비밀 키 집합에서 암호학적 해시 함수 H를 통해 서명하고자 하는 메시지에 따라 서로 다른 k개의 원소를 선택하여 이들로 서명을 구성한다. HORSIC도 HORS와 같은 방법으로 암호학적 해시 함수 H를 통해 k개의 원소를 선택하지만, 그 후 또 다른 암호학적 해시 함수 G와 전단사 함수 $C_{k,z}$를 통해 서명하고자 하는 메시지에 따라 서로 다른 횟수의 일방향 함수를 앞에서 선택한 k개의 원소 각각에 적용하여 이들로 서명을 구성한다. 따라서 HORS에 비해 서명 위조 확률을 줄일 수 있다.

HORSIC을 제외한 기존 일회용 서명 기법들의 서명 위조 확률은 공개 키 크기와 서명 크기에 의해 결정된다. 따라서 이러한 기법들의 경우 보안성을 낮추지 않으면서 공개 키 크기와 서명 크기를 동시에 줄이는 것은 불가능하다. 하지만 HORSIC은 서명하고자 하는 메시지에 따라 비밀 키에 적용되는 일방향 함수의 횟수를 서로 다르게 함으로써 서명 위조 확률을 줄였으며, 따라서 기존 일회용 서명 기법들에 비해 작은 크기의 공개 키나 서명을 사용하면서도 같은 수준의 보안성을 제공할 수 있다. 대신 키 생성과 서명 검증 비용은 기존 일회용 서명 기법들에 비해 큰 편이다. 하지만 센서 노드에서 전체적으로 소모되는 에너지의 양은 HORSIC의 경우 서명의 크기가 줄어듬으로 인해 다른 일회용 서명 기법들에 비해 오히려 더 적은 경우가 많다.

비록 HORSIC이 무선 센서 네트워크에서 브로드캐스트 인증을 위해 설계된 일회용 서명 기법이기는 하지만, 스마트그리드와 같이 제한된 자원을 가진 노드들로 이루어진 다른 다양한 네트워크에서도 사용 가능하다.

참 고 문 헌

[AK04] Ian F. Akyildiz and Ismail H. Kasimoglu. Wireless sensor and actor networks: research challenges. *Ad Hoc Networks*, 2(4):351–367, 2004.

[And98] G.E. Andrews. *The theory of partitions*, volume 2. Cambridge University Press, 1998.

[ASSC02] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. Wireless sensor networks: a survey. *Computer networks*, 38(4):393–422, 2002.

[BDE+11] J. Buchmann, E. Dahmen, S. Ereth, A. Hülsing, and M. Rückert. On the security of the winternitz one-time signature scheme. *Progress in Cryptology – AFRICACRYPT 2011*, pages 363–378, 2011.

[BDF01] Dan Boneh, Glenn Durfee, and Matt Franklin. Lower bounds for multicast message authentication. *Advances in Cryptology - EUROCRYPT 2001*, pages 437–452, 2001.

[BDK+07] J. Buchmann, E. Dahmen, E. Klintsevich, K. Okeya, and C. Vuillaume. Merkle signatures with virtually unlimited signature capacity. In *Applied Cryptography and Network Security*, pages 31–45. Springer, 2007.

[BGD+06] J. Buchmann, L. García, E. Dahmen, M. Döring, and E. Klintsevich. Cmss – an improved merkle signature scheme. *Progress in Cryptology - INDOCRYPT 2006*, pages 349–363, 2006.

[BM94] Daniel Bleichenbacher and Ueli M. Maurer. Directed acyclic graphs, one-way functions and digital signatures. In *Advances in Cryptology - CRYPTO '94, volume 839 of Lecture Notes in Computer Science*, pages 75–82. Springer-Verlag, 1994.

[BM96] Daniel Bleichenbacher and Ueli M. Maurer. Optimal tree-based one-time digital signature schemes. *In STACS' 96: Proceedings of the 13th Annual Symposium on Theoretical Aspects of Computer Science*, 1046:361–374, 1996.

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *Proceedings of the 1st ACM conference on Computer and communications security*, pages 62–73. ACM, 1993.

[CJ03] Don Coppersmith and Markus Jakobsson. Almost optimal hash sequence traversal. In *Financial Cryptography*, pages 102–119. Springer, 2003.

[CSLH06] Shang-Ming Chang, Shiuhpyng Shieh, Warren W. Lin, and Chih-Ming Hsieh. An efficient broadcast authentication scheme in wireless sensor networks. In *Proceedings of the 2006 ACM Symposium on Information, computer and communications security*, ASIACCS '06, pages 311–320, New York, NY, USA, 2006.

[DH76] W. Diffie and M. Hellman. New directions in cryptography. *Information Theory, IEEE Transactions on*, 22(6):644–654, 1976.

[DHHM05] J. Deng, C. Hartung, R. Han, and S. Mishra. A practical study of transitory master key establishment for wireless sensor networks. In *Security and Privacy for Emerging Areas in Communications Networks, 2005. SecureComm 2005. First International Conference on*, pages 289–302. IEEE, 2005.

[EGE02] Jeremy Elson, Lewis Girod, and Deborah Estrin. Fine-grained network time synchronization using reference broadcasts. *SIGOPS Oper. Syst. Rev.*, 36(SI):147–163, December 2002.

[EGM89] Shimon Even, Oded Goldreich, and Silvio Micali. On-line/off-line digital signatures. In *Proceedings on Advances in cryptology*, CRYPTO '89, pages 263–275, New York, NY, USA, 1989. Springer-Verlag New York, Inc.

[ER03] Jeremy Elson and Kay Römer. Wireless sensor networks: a new regime for time synchronization. *ACM SIGCOMM Computer Communication Review*, 33(1):149–154, January 2003.

[GPW+04] Nils Gura, Arun Patel, Arvinderpal Wander, Hans Eberle, and Sheueling Chang Shantz. Comparing elliptic curve cryptography and rsa on 8-bit cpus. *Cryptographic Hardware and Embedded Systems-CHES 2004*, pages 925–943, 2004.

[Gro96] Lov K. Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, STOC '96, pages 212–219, New York, NY, USA, 1996.

[HBH05] C. Hartung, J. Balasalle, and R. Han. Node compromise in sensor networks: The need for secure systems. *University of Colorado at Boulder, Technical Report TR-CU-CS-990-05*, 2005.

[HE03] Lingxuan Hu and D. Evans. Secure aggregation for wireless networks. In *Applications and the Internet Workshops, 2003. Proceedings. 2003 Symposium on*, pages 384–391. IEEE, 2003.

[HSW+00] Jason Hill, Robert Szewczyk, Alec Woo, Seth Hollar, David Culler, and Kristofer Pister. System architecture directions for networked sensors. *SIGPLAN Not.*, 35(11):93–104, November 2000.

[KKP99] J. M. Kahn, R. H. Katz, and K. S. J. Pister. Next century challenges: mobile networking for "smart dust". In *Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*, MobiCom '99, pages 271–278, New York, NY, USA, 1999. ACM.

[KL08] Jonathan Katz and Yehuda Lindell. *Introduction to modern cryptography*. Chapman & Hall, 2008.

[Koc94] C.K. Koc. High-speed rsa implementation. Technical report, RSA Laboratories, 1994.

[KSW04] Chris Karlof, Naveen Sastry, and David Wagner. Tinysec: a link layer security architecture for wireless sensor networks. In *Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 162–175. ACM, 2004.

[KW03] Chris Karlof and David Wagner. Secure routing in wireless sensor networks: attacks and countermeasures. *Ad Hoc Networks*, 1(2–3):293–315, 2003.

[Lam79] Leslie Lamport. Constructing digital signatures from a one-way function. Technical report, SRI-CSL-98, SRI International Computer Science Laboratory, October 1979.

[LC11] Qinghua Li and Guohong Cao. Multicast authentication in the smart grid with one-time signature. *Smart Grid, IEEE Transactions on*, 2(4):686–696, 2011.

[LG09] G. Locke and P. Gallagher. Fips pub 186-3: Digital signature standard (dss). *National Institute of Standards and Technology*, 2009.

[LKC+12] Jaeheung Lee, Seokhyun Kim, Yookun Cho, Yoojin Chung, and Yongsu Park. Horsic: An efficient one-time signature scheme for wireless sensor networks. *Information Processing Letters*, 112(20):783–787, 2012.

[LPW06] Mark Luk, Adrian Perrig, and Bram Whillock. Seven cardinal properties of sensor network broadcast authentication. In *Proceedings of the fourth ACM workshop on Security of ad hoc and sensor networks*, pages 147–156. ACM, 2006.

[Mer79] Ralph Merkle. *Secrecy, authentication, and public key systems.* PhD thesis, Stanford, CA, USA, 1979.

[Mer88] Ralph Merkle. A digital signature based on a conventional encryption function. In *A Conference on the Theory and Applications of Cryptographic Techniques on Advances in Cryptology*, CRYPTO '87, pages 369–378, London, UK, UK, 1988. Springer-Verlag.

[Mer89] Ralph Merkle. A certified digital signature. In *Proceedings on Advances in cryptology*, CRYPTO '89, pages 218–238, New York, NY, USA, 1989. Springer-Verlag New York, Inc.

[MMO85] S.M. Matyas, C.H. Meyer, and J. Oseas. Generating strong one-way functions with cryptographic algorithm. *IBM Technical Disclosure Bulletin*, 27(10A):5658–5659, 1985.

[MOI90] S. Miyaguchi, K. Ohta, and M. Iwata. 128-bit hash function (n-hash). *NTT review*, 2(6):128–132, 1990.

[MP02] M. Mitzenmacher and A. Perrig. Bounds and improvements for biba signature schemes. *No. TR-02-02, Computer Science Group, Harvard University, USA*, 2002.

[MVOV97] A.J. Menezes, P.C. Van Oorschot, and S.A. Vanstone. *Handbook of applied cryptography*. CRC, 1997.

[MWS04] D.J. Malan, M. Welsh, and M.D. Smith. A public-key infrastructure for key distribution in tinyos based on elliptic curve cryptography. In *Sensor and Ad Hoc Communications and Networks, 2004. IEEE SECON 2004. 2004 First Annual IEEE Communications Society Conference on*, pages 71–80. IEEE, 2004.

[NCG02] M.A. Nielsen, I. Chuang, and L.K. Grover. Quantum computation and quantum information. *American Journal of Physics*, 70:558, 2002.

[NIS02] NIST. Fips pub 180-2: Secure hash standard (shs). *National Institute of Standards and Technology*, 2002.

[PC06] Yongsu Park and Yookun Cho. Efficient one-time signature schemes for stream authentication. *Journal of information science and engineering*, 22(3):611–624, 2006.

[PCST01] Adrian Perrig, Ran Canetti, Dawn Song, and J. D. Tygar. Efficient and secure source authentication for multicast. In *Proceedings of the*

Internet Society Network and Distributed System Security Symposium (NDSS 2001), pages 35–46, 2001.

[Per01] Adrian Perrig. The biba one-time signature and broadcast authentication protocol. In *Proceedings of the 8th ACM conference on Computer and Communications Security*, pages 28–37. ACM, 2001.

[PGV94] Bart Preneel, René Govaerts, and Joos Vandewalle. Hash functions based on block ciphers: A synthetic approach. In *Proceedings of the 13th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '93, pages 368–378, London, UK, 1994. Springer-Verlag.

[PLP06] Krzysztof Piotrowski, Peter Langendoerfer, and Steffen Peter. How public key cryptography influences wireless sensor node lifetime. In *Proceedings of the fourth ACM workshop on Security of ad hoc and sensor networks*, pages 169–176. ACM, 2006.

[Pre93] Bart Preneel. *Analysis and design of cryptographic hash functions*. PhD thesis, Katholieke Universiteit te Leuven, 1993.

[PSP03] Bartosz Przydatek, Dawn Song, and Adrian Perrig. Sia: secure information aggregation in sensor networks. In *Proceedings of the 1st international conference on Embedded networked sensor systems*, SenSys '03, pages 255–265, New York, NY, USA, 2003. ACM.

[PST+02] Adrian Perrig, Robert Szewczyk, J. D. Tygar, Victor Wen, and David E. Culler. Spins: Security protocols for sensor networks. *Wireless networks*, 8(5):521–534, 2002.

[PTSC00] Adrian Perrig, J. D. Tygar, Dawn Song, and Ran Canetti. Efficient authentication and signing of multicast streams over lossy channels. In *Proceedings of the 2000 IEEE Symposium on Security and Privacy*, pages 56–73. IEEE, 2000.

[Rab78] M.O. Rabin. Digitalized signatures. *Foundations of Secure Computation*, 78:155–166, 1978.

[Riv92] R. Rivest. The md5 message-digest algorithm. *Internet Request For Comments 1321*, 1992.

[Roh99] Pankaj Rohatgi. A compact and fast hybrid signature scheme for multicast packet authentication. In *Proceedings of the 6th ACM conference on Computer and communications security*, CCS '99, pages 93–100, New York, NY, USA, 1999.

[RR02] Leonid Reyzin and Natan Reyzin. Better than biba: Short one-time signatures with fast signing and verifying. In *Information Security and Privacy*, pages 1–47. Springer, 2002.

[RSA78] R.L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

[Sho99] Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM review*, 41(2):303–332, June 1999.

[SOS+08] Piotr Szczechowiak, Leonardo Oliveira, Michael Scott, Martin Collier, and Ricardo Dahab. Nanoecc: Testing the limits of elliptic curve cryptography in sensor networks. *Wireless sensor networks*, pages 305–320, 2008.

[WGE+05] A.S. Wander, N. Gura, H. Eberle, V. Gupta, and S.C. Shantz. Energy analysis of public-key cryptography for wireless sensor networks. In *Pervasive Computing and Communications, 2005. PerCom 2005. Third IEEE International Conference on*, pages 324–328. IEEE, 2005.

Abstract

Broadcast authentication is a basic and important security primitive in wireless sensor networks. To provide broadcast authentication in wireless sensor networks, several approaches have been proposed so far. However, most of them are not suitable for wireless sensor networks. For example, to use digital signature schemes based on public key cryptography is considered to be too computation intensive for resource constrained sensor nodes. μTESLA requires loose time synchronization between the nodes and causes an authentication delay.

Broadcast authentication can be achieved using one-time signature schemes. Since one-time signature schemes are based on one-way functions without trapdoors, one-time signature schemes are very efficient in computations. However, one-time signature schemes require a large amount of storage space and have a large signature size resulting in a high communication overhead.

This dissertation proposes HORSIC, an efficient one-time signature scheme for wireless sensor networks. HORSIC is basically an extension of HORS. HORS uses only a cryptographic hash function H to make it infeasible to find two distinct messages that will result in the same k-element subset. HORSIC decreases the probability of forgery by using another cryptographic hash function G and a bijective function $C_{k,z}$ as well as H to make it infeasible to find two distinct messages that

will result in the same k-part integer composition as well as the same k-element subset.

For all one-time signature schemes except HORSIC, the probability of forgery is determined only by the public key size and the signature size. Thus, in this case, both the public key size and the signature size cannot be decreased simultaneously. However, HORSIC can reduce the public key size and the signature size simultaneously by sacrificing key generation and verification costs; HORSIC iteratively applies a one-way permutation on secret values, while the number of iterations depends on the message signed. Even though the verification cost of HORSIC is higher than that of other OTS schemes, the computation energy consumption caused by the increased verification cost can be more than compensated for by the communication energy consumption caused by the reduced signature size.

While designed primarily for providing broadcast authentication in wireless sensor networks, HORSIC can also be used for other networks, such as a smart grid consisting of nodes with limited resources.

Keywords: Wireless Sensor Networks, Cryptography, Authentication, Broadcast Authentication, One-time Signature

Student Number: 2007-30237

감 사 의 글

Life is 10 percent what you make it, and 90 percent how you take it.

— Irving Berlin

어느덧 6년 간의 박사과정을 마무리하며 지난 시간들을 돌이켜보니 많은 분들의 격려와 도움이 있었습니다.

먼저 본 논문이 완성되기까지 세심한 지도와 많은 격려로 이끌어 주신 조유근 교수님께 진심으로 감사드립니다. 또한 바쁘신 와중에도 심사 위원으로서 이 논문의 부족함을 채워 주신 신현식 교수님, 박근수 교수님, 민상렬 교수님 그리고 박용수 교수님께도 감사드립니다.

박사과정 동안 같이 지낸 시스템 소프트웨어 연구실 동기 및 선후배님들께도 감사드립니다. 제은태 실장님, 홍지만 교수님, 정유진 교수님, 전광일 교수님, 상수 형, 진혁 형, 희헌 형, 근영 형, 수관 형, 준영 형, 상호, 학봉 형, 본철, 홍, 용태, 석현 형, 승우, 진하, 경동, 진만 형, 봉재, 상일, 준혁, 지연, 철, 승환 씨, 동화 모두 감사드립니다. 각자의 분야에서 좋은 일들만 있기를 진심으로 기원합니다.

오랜 친구인 상근이와 기영이에게도 고마움을 전합니다. 자주 보지는 못했지만 지치고 힘들 때마다 언제나 큰 힘이 되어 주었습니다.

마지막으로, 항상 옆에서 힘이 되어주는 가족들에게 감사드립니다. 부끄럽지 않도록 앞으로도 열심히 살겠습니다. 감사합니다.