유비쿼터스 환경에서 해쉬 락 기법을 적용한 보안과 프라이버시에 관한 연구

A Study of Security and Privacy and using Hash Lock Approach in Ubiquitous environment

최용식, Yongsik Chio*, 전영준, Youngjun John*, 박상현, Sanghyun Park*, 한수, Soo Han*, 신승호, Sungho Shin*
*인천대학교 컴퓨터공학과

요약 최근 유비쿼터스 컴퓨팅에 대한 연구가 활발히 진행되고 있으며 유비쿼터스 컴퓨팅의 실현을 위한 핵심기술로서 RFID 시스템에 대한 연구가 활발히 진행되고 있다. 유비쿼터스 환경에서 RFID 시스템이 사용자의 편리함을 가져다 주는 장점이 있는 반면, 이로 인해 사용자의 프라이버시가 침해 당할 수 있는 문제점 또한 가지고 있다. 본 논문에서 사용자 인증 알고리즘은 새로운 해쉬 함수를 사용하고 그리고 메시지 암호화를 위한 스트림 암호기는 LFSR(Linear Feedback Shift Register)을 사용한다.

핵심어: 유비쿼터스 컴퓨팅, 해쉬 함수, 해쉬 락, LFSR(Linear Feedback Shift Register)

1. 서론

유비쿼터스 컴퓨팅(Ubiquitous Computing)은 새로운 IT 패러다음으로 1988년 미국 제록스 팰로앨토연구소의 마크 와이저(Mark Weiser)가 제안한 개념이다. 이는 일상생활과 컴퓨팅이 접목된 지능화된 환경을 통해 제한 없이 접속하고 쉽게 서비스를 제공받을 수 있도록 발전되었으며, 기술의 비약적 발전을 통해 실현 가능성을 높여지고 있다. 또한 유비쿼터스 컴퓨팅 환경은 네트워크를 기반으로 한 장치간의 연결을 기본으로 하고 있다. 특히, 무선중심의 근거리 통신기술이 발달함에 따라 유비쿼터스 컴퓨팅 환경을 이루는 무수히 많은 개체들은 유기적으로 연결되어, 서로 데이터를 주고받고, 이를 통해 서비스를 제공하게 된다. 이에 따른 정보보호 관점의 보안 요구사항이 도출될 수 있다.

본 논문에서는 이러한 유비쿼터스 컴퓨팅 환경 내에서 발생할 수 있는 보안 취약점과 보안 요구사항을 도출하여 제시하고, 인증기술을 중심으로 한 보안 서비스 방향을 제안함으로써 센서네트워크 및 RFID(Radion Frequency Identification) 기술의 현실화될 수 있는 기반기술인 보안 및 인증기술에 대해 연구하고자 한다. 이를 위해서는 정보의 무결성 확인과 메시지 인증 코드, 디지털 서명의 효율성 증대 등 현대 암호학에서 중요한 역할을 수행하고 있는 해쉬 락 기법(Hash Lock Approach)을 제안하고자 한다. 이는 Tag에서 수행하는 기능은 단지 해쉬 기능만으로 보안 서비스가 가능한 스킴으로써 각 Tag는 Reader와 공유되는 Key를 보유하고 있으며 초기 Tag 식별 정보로 Key를 해쉬한 Meta ID 정보만을 제공함으로 실제적인 ID 정보의 유통을 방지하는 메커니즘이다. 이는 일방향 해시 함수의 역함수 계산 어려움에 기반한 해쉬 락에 PKI(Public Key Infrastructure)방법을 적용하여 MetaID를 비밀키로써 사용한다. Reader는 미리 등록된 공개키(meta ID를 이용하여 생성된)로 Tag를 인증하고 meta ID로 각 Tag의 유일한 키(k)를 생성하며 이에 해당하는 meta ID = H(k)를 가지고 있다. 이 때 H( )는 해쉬함수이다. Tag는 자신의 비밀키를 이용하여 생성된 meta ID를 Reader에 보내고 Reader는 해당되는 키(k)를 만들어내고 Tag에 보낸다. 이때 Tag는 Reader로부터 보내어진 키(k)를 해쉬값과 자신의 meta ID를 비교하여, 그 값이 동일하면 자신의 ID를 전송한다.

2. 관련연구

유비쿼터스 환경에서 안전하지 않는 RFID Tag를 사용하는 경우 물리적 공격, 위조, Spoofing, 도청, 트래픽 분석, DOS 공격 등에 의한 보안적 취약점에 노출된다. 이러한 취약점은 개인 및 조직의 보안이나 프라이버시에 매우 중요한 영향력을 갖는다. RFID를 위한 컴퓨팅 환경은 일반적인 인터넷 환경과는 달리 많은 제약적 사항을 갖는다. 이러한 제약적 사항은 Cellular

Phone 등을 이용한 무선 인터넷보다 더욱 자원 측면적 한계를 위해서는 모든 상품이나, 사람 등 객체에 설치되는 Tag 가격은 저렴하게 구현되어야 하며 대신에 Reader 장비나 Back End 시스템에서 많은 성능, 자원 측면에서 열악한 Tag 장비의 자원적 한계를 극복할 수 있도록 설계, 운영되어야 한다. 이에 보안 기술 적용에 대한 부분도 이러한 운영, 환경 측면을 충분히 고려해야 한다.

2.1 USN(Ubiquitous Sensor Network) 보안기술

유비쿼터스 네트워크 연결은 장소나 시간에 제약 없이 언제나 최상의 서비스를 받기 위한 편리성을 제공할 것이다. RFID 기술은 이러한 유비쿼터스의 목적을 쉽게 이룰 수 있는 기술로 현재 주목을 받고 있으며 많은 장점을 갖는다. 이러한 RFID 환경에서 가장 보안적인 문제를 갖는 것은 RFID Tag에 저장 관리되는 식별 정보에 대한 프라이버시 침해 문제를 어떻게 보호할 것인가 이다.

RFID 환경에서 안전하지 않는 Tag를 사용하는 경우 물리적 공격, 위조, Spoofing, 도청, 트래픽 분석, DOS 공격 등에 의한 보안적 취약점에 노출된다. 이러한 취약점은 개인 및 조직의 보안이나 프라이버시에 매우 중요한 영향력을 갖는다. RFID를 위한 컴퓨팅 환경은 일반적인 인터넷 환경과는 달리 많은 제약적 사항을 갖는다. 이러한 제약적 사항은 Cellular Phone 등을 이용한 무선 인터넷보다 더욱 자원 측면적 한계를 위해서는 모든 상품이나, 사람 등 객체에 설치되는 Tag 가격은 저렴하게 구현되어야 하며 대신에 Reader 장비나 Back End 시스템에서 많은 성능, 자원 측면에서 열악한 Tag 장비의 자원적 한계를 극복할 수 있도록 설계, 운영되어야 한다. 이에 보안 기술 적용에 대한 부분도 이러한 운영, 환경 측면을 충분히 고려해야 한다. RFID 시스템은 3개의 구성요소를 갖는다.

① RFID Tag(Transponder) : 객체 식별 정보 전송

② RFID Reader(Transceiver) : Tag 정보 read/write

③ Back End DB : Reader에 의해 수집되는 Tag 확인 정보 제공

RFID 시스템 내 식별되어지기 위한 모든 객체는 Tag와 함께 물리적 라벨을 갖는다. Tag는 전형적으로 저장하기 위한 마이크로 칩과 프로세서, 안테나 등으로 구성된다. 위에서 언급한 다양한 보안적 요구사항을 만족하기 위해서는 현재의 일반적인 RFID 환경에 대한 제약 사항을 고려해야 한다. RFID 환경에 대한 제약 사항은 다음과 같다.

① RFID Type : RFID Tag 장비는 자체적인 전원을 갖는 능동형 Tag와 자체적인 전원을 갖지 않는 수동형 Tag가 존재한다. 일반적인 환경에 설치, 운영되어지기 위해서는 수동형 Tag를 선호한다.

② 전송 데이터 크기 : 현재 전송 대역에 대한 표준 기술로 13.56Mhz와 900Mhz로 구분되어지며 13.56Mhz는 26Kbps/50tag를 동시에 처리할 수 있으며 900Mhz는 128Kbps/200tag를 처리할 수 있는 용량을 갖는다. Tag Reader 장비에서 각 Tag 판독을 처리하기 위한 시간은 1초를 초과하지 말아야 하며 각 Tag는 500비트의 전송 데이터를 갖도록 한다.

③ Tamper Resistant : Tag에 대한 물리적 공격에 자체적 정보 보호(예: 자동 정보 삭제 등)를 Tag 주요 기능으로 부가하기에는 가격적인 측면에서 부적절하다. 따라서 Tag는 tamper resistant가 제공되어지지 않는다고 전제해야 한다.

④ 쓰기 제약 : Tag 메모리에 쓰기는 Tag와 Writing 장비 간에 거리, 장비 타입, 특정 명령어, 패스워드 등에 대한 제약을 가질 수 있다.

2.2 데이터 인증(Data Authentication)

2.2.1 Kill Command

MIT 대학을 중심으로 하는 Auto ID Center에 의해 제안한 기술로서 각 Tag는 유일한 8bit의 패스워드를 갖고 특정 지역의 Reader 장비에 의해 읽혔을 경우 패스워드가 지워짐으로서 프라이버시에 대한 보호를 목적으로 한다. 그러나 kill command가 적절한 위치에서 적절하게 기능을 수행하는 것에 대한 보장을 하기가 어려우며 또한 삭제된 Tag 정보의 되살리기가 어렵기 때문에 많은 RFID Tag 장점을 활용할 수 없는 단점을 갖는다.

2.2.2 Anonymous ID

NTT에서 제안한 기술로서 Tag 주요 식별 정보는 공개키나 대칭키 등을 이용하여 암호화하여 Tag에 주입하고 이 값을 Reader 장비로 전송함으로 인가된 Reader 장비만이 식별 정보를 복호화하여 쓸 수 있는 메커니즘이다. 전제 사항으로 Reader 장비와 Back End Server 간에 보안 채널을 보장해야 한다. 암호화된 ID 정보 역시 고정 값으로서 의도적으로 암호화된 ID 정보의 변경이 가해져야 하나 실제적인 구현에 한계를 갖는다.

2.2.3 External re-encryption

RSA 사에서 제안하는 기술로서 공개키 기술을 이용하는 것이다. 즉 Tag에 저장되는 암호화 ID 값은 매번 Reader와 같은 외부 장비에 의해 암호화 되고 반복해서 Tag에 저장하는 매커니즘이다. 공개키 값의 변화에 따라 암호화 값이 자주 변경되어짐으로 공격자의 추적이 용이하지 않다. 그러나 Tag의 반복적 쓰기는 현실적인 환경에서는 아직 가능하지 않다.

2.2.4 XOR based on time pad

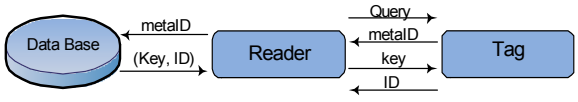
역시 RSA사에서 제안하는 기술로서 Tag와 Back End Server(Reader) 간에 공유하는 난수 키 목록을 보유하고 있고 양단에 난수 키에 대한 비교 검증이 이루어지면 Tag가 ID를 전송하는 구조를 갖는다. 비교 시 단순한 XOR 기능만을 Tag가 수행함으로 성능적인 우수성을 갖는다. 그러나 양단 간 몇 번에 걸친 통신이 이루어져야 하며 매번 난수 키 목록을 업데이트하

는 부분에서 문제점이 존재한다.

2.3 Hash Lock Approach와 Randomized Hash Lock Approach

2.3.1 Hash lock 방식

일방향 해시 함수의 역함수 계산 어려움에 기반한 Hash Lock 방식은 인가받지 않은 Reader기가 Tag를 읽는 것을 방지 할 수 있다. Spoofing은 방지하지 못하지만 탐지는 가능하다. 이 방식은 해쉬 함수만을 요구하므로 저비용으로 구현될 수 있으나, meta ID가 고정되어 공격자는 meta ID를 이용하여 해당 Tag의 위치를 추적할 수 있는 문제가 있다. [3,4].

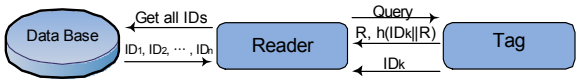


[그림 1] Hash Lock의 Unlocking 프로토콜

[그림 1]와 같이 Reader는 Tag에게 MetaID를 질의하고 DataBase는 (MetaID, Key)를 조사하여 Reader는 Tag에게 Key를 전송한다. 만약 Hash(key)와 MetaID가 일치하면 잠긴 상태에서 빠져 나온다.

2.3.2 Randomized hash lock 방식

이 방식은 Hash lock 방식을 개선한 것으로 고정된 meta ID를 갖게 하지 않기 위하여 난수 생성기를 통하여 접근할 때마다 Tag에서 다른 출력 값을 가지게 한다. 그러므로 Tag에 대한 추적은 불가능하다. 그러나 해쉬 함수와 난수가 동시에 생성되어야 하므로 저비용으로 구현하기 어렵다. [3,4].



[그림 2] Randomized Hash Lock의 Unlocking 프로토콜

[그림 2]와 같이 Reader는 Tag에게 질의를 보낸다. Tag는 랜덤한 난수를 생성하고 Hash(ID || R) 값을 계산하고 Tag는 Reader에게 전송하여 Hash(ID_i || R) == Hash(ID || R)을 만족하는 ID_i를 찾으면 전송하고 잠긴 상태에서 빠져 나온다.

3. 해쉬 함수 알고리즘

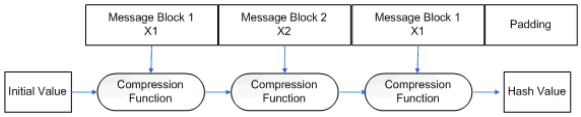
3.1 해쉬 함수 정의

암호학적 해쉬 함수(Cryptographic hash functions)는 정보의 무결성 확인과 메시지 인증 코드, 디지털 서명의 효율성 증대 등 현대 암호학에서 중요한 역할을 수행하고 있다. 이는 비암호학적 컴퓨터 응용분야에 사용되고 있는 일반 해쉬 함수와 비교할 때, 두 경우 모두 큰 정의역에서 작은 치역으로의 함수이지만 몇 가지 중요한 측면에서 서로 다르다. 또한 해쉬 함수는 임의의 길이를 갖는 메시지를 입력으로 하여 고정된 길이의 해쉬 값(hash value) 또는 해쉬 코드(hash code)라 불리는 값을 출력한다. 보다 엄밀하게 말하면 해쉬 함수 h는 임의의 길이의 문자열을 고정된 길이를 갖는 n비트 문자열로 대응시킨다. 정의역을 D, 치역을 R이라 할 때 해쉬 함수 는 다대일(may-to-one) 대응 함수이다. 따라서 일반적으로 해쉬 함수는 충돌(collision)이 반드시 존재한다. 예를 들어 해쉬 함수 가 비트의 입력 값과 비트의 출력 값을 갖는 함수라 가정하면, 가 임의성을 가질 때 개의 엽력 값이 각각의 출력 값에 대응한다. 따라서 확률로 두 개의 임의로 선택된 입력 값은 의 값과는 무관하게 같은 출력 값을 가진다.

거의 모든 해쉬함수의 처리 과정은 입력을 연속적인 고정된 블록들로 나누어 처리함으로써 임의의 길이를 갖는 반복적인 처리 과정이다. 입력 X는 블록 길이의 배수가 되도록 패딩(Padding)되고 t개의 블록 X_1에서 X_t로 나누어 진다. 해수함수 h는 다음과 같다.

$$H_0 = IV$$
$$H_i = f(H_{i-1}, X_i), 1 \le i \le t$$
$$h(X) = H_t$$

여기서 f는 암축함수(compress function)이며 Hi는 단계 i-1과 단계 i사이에서 연쇄 변수(chaining variable)이고 IV는 초기값(initial value)이다. 압축함수를 사용한 반복적인 해쉬함수의 일반적인 구조는 [그림 3]와 같다.



[그림 3] 반복적인 해쉬함수의 일반적인 구조

해쉬값의 계산은 연쇄 변수에 의존한다. 해쉬 계산을 시작할 때, 이 연쇄 변수는 알고리즘의 일부로 명시된 고정된 초기값을 가진다. 압축 함수는 해쉬 될 메시지 블록을 입력으로 받아 이 연쇄 변수의 값을 갱신한다. 이 과정이 모든 메시지 블록에 대해 순환적으로 반복되고, 연쇄 변수의 마지막 값이 그 메시지에 대한 해쉬값으로 출력된다. 해쉬함수는 내부 함수로 어떤 구조를 사용하냐에 따라서 블록 암호 기반 해쉬함수, 모듈러 연산 기반 해쉬함수, 전용 해쉬함수로 분류된다. 전용 해쉬함수는 빠른 처리 속도를 가지고 다른 시스템 서브 요소에 무관하도록 해쉉을 위해 특별히 설계된 함수들이다. 현째까지 제안된 전용해쉬함수는 대부분 1990년 Rivest에 의해 설계된 MD4[3]에 기반한 구조를 가진다. 현재 널리 사용되는 MD계열 해쉬함수로는 MD5[4], SHA-1[5], RIPEMD-160[6], HAVAL[7]등이 있다.

특정 해쉬함수가 주어지면, 안전한 해쉬함수의 입증

을 위해 해쉿함수를 공격하는 복잡도에 관한 하한을 증명할 수 있는 것이 바람직하지만 실제 그러한 방법은 거의 알려져 있지 않고 대부분의 경우에 적용 가능한 알려진 공격의 복잡도가 해쉬함수의 안전성으로 고려된다. 해쉬값이 균등한 확률 변수라고 가정하면, 다음은 잘 알려진 사실이다[1].

본 논문에서는 인증을 위한 알고리즘은 GSM(Global System for Mobile Communication)에서 표준화되어 있지 않은 관계로 통신망 관리자가 적당한 알고리즘을 선택하여야 한다. PLMN(Public Land Mobile Network)의 보안 레벨은 통신망 관리자의 보안 인식에 달려 있으며, 인증 알고리즘은 고비도의 단방향 함수로서 주어진 RAND(Random Number)와 SRES(Signed Response)로부터 인증키를 유출하기가 불가능하도록 설계되어야 한다. 알고리즘의 입출력 조건은 RAND가 128비트이고 SRES는 32비트로 표준화되어 있다. 인증 센터는 인증 파리미터 RAND와 SRES를 발생하고 각 가입자가 등록된 HPLMN(Home PLMN) 또는 VPLMN(Visited PLMN)에서 이루어진다. 이것은 인증 파라미터가 수시로 타지역 PLMN에 전달되어야 함을 의미한다.

암호키 생성의 경우는 데이터의 암/복호화를 위해 사용되는 암호키 생성과정은 인증 절차가 완료된 후에 생성되며, 키 변경 주기는 인증 빈도수에 따르게 되며 통신망 관리자에 의해 결정된다. 본 논문에 사용되는 인증과 암호키 생성 알고리즘은 동일한 입력 파라미터 RAND와 를 사용하여 암호키를 생성한다. 암호키가 무선 채널에 노출되지 않도록 하기 위해 인증 센터와 스마트 카드에 실장된 알고리즘으로 생성하여 사용한다. 또한 암호키 생성 알고리즘은 GSM에서 표준화되지 않았지만 외부 파라미터가 인증을 위한 알고리즘 방법과 동일하므로 단일 알고리즘으로 구성하도록 권고되고 있다.

3.2 제안 알고리즘

본 논문에서 제안한 전용 해쉬 함수는 32비트 프로세스에서 설계되었으며 CPU 기본 연산인 덧셈, 뺄셈, 곱셈, 배타적 논리합 연산을 사용하였다. MD계열 전용 해쉬함수의 경우 비선형성을 높이기 위하여 부울함수를 사용하였으나 본 논문에서는 x^{-1} 연산을 사용하였다. 일반적으로 역원을 구하는 연산은 시간이 많이 걸리나 $GF(2^8)$ 에서의 역원이므로 미리 연산하여 참조테이블을 구성하였다. 모든 연산은 32비트 단위로 이루어지며 6개의 32비트 레지스터 a, b, c, d, e, f는 연쇄변수로서 최종 해쉬값을 갖는다. 이 레지스터들은 값으로 초기화되는데 그 값은 다음과 같다.

a = 0x01234567; b = 0xefcdab89; c = 0x98badcef;

d = 0x10325476; e = 0xc3d2e1f0; f = 0x5a3cf01d;

256 비트 메시지 블록은 32비트 단위로 나뉘어져 x0, x1, x2 …… x7 레지스터의 초기값으로 사용되고 이어지는 연산에 의해 는 로 갱신된다. 따라서 256비트 메시지는 3번의 패스로 이루어지며 그 사이에 키 스케쥴링이 이루어지며, 최종적으로 feedforward 단계에서는 레지스터 a, b, c, d, e, f의 현재값 을 만들어낸다.

3.2.1 알고리즘 구조

save_abcdef

pass(a, b, c, d, e, f, 3)

key_schedule

pass(a, b, c, d, e, f, 5)

key_schedule

pass(a, b, c, d, e, f, 7)

feedforward

① save_abcdef는 feedforward 단계에서 사용할 h_i 초기 값을 저장한다.

aa = a; bb = b; cc = c; dd = d; ee = e; ff = f;

② pass(a, b, c, d, e, f, mul)은 다음과 같이 구성된다.

round(a, b, c, d, e, f, x0, mul);

round(b, c, d, e, f, a, x1, mul);

round(c, d, e, f, a, b, x2, mul);

round(d, e, f, a, b, c, x3, mul);

round(e, f, a, b, c, d, x4, mul);

round(f, a, b, c, d, e, x5, mul);

round(a, b, c, d, e, f, x6, mul);

round(f, b, d, a, c, e, x7, mul);

round(a, b, c, d, e, f, X, mul)은 다음과 같다.

f ^= X;

a -= Gen_32(f, f, f, f);

f ^= a;

b + = Gen_32(f, f, f, f);

b *= mul;

f ^= b;

c + = Gen_32(f, f, f, f);

c *= mul;

f ^= c;

d + = Gen_32(f, f, f, f);

d *= mul;

f ^= d;

e + = Gen_32(f, f, f, f);

e *=mul;

여기서 Gen_32( ) 함수는 32비트 레지스터 4개를 입력으로 받아 각 레지스터의 첫 번째, 두 번째, 세 번째, 네 번째 8비트를 S-box의 입력으로 사용하고 이에 대응하는 S-box 출력을 가지고 32비트 값을 만드는 함수이다.

③ key_schedule은

x0 -= x7 ^ 0xA5A5A5A5;

x1 ^= x0;

x2 + = x1;

x3 -= x2; ^ ((~ x1) << 7);

x4 ^= x3;

x5 + = x4;

x6 -= x5 ^((~ x4) >> 23);

x7 ^= x6;

x0 + = x7;

x1 -= x0 ^((~ x7) << 7);

x2 ^= x1;

x3 + = x2;

x4 -= x3 ^((~ x2) >> 23);

x5 ^= x4;

x6 + = x5;

x7 ^= x6 ^ 0x01234567;

다음과 같다. 여기서 >>, << 은 좌, 우 논리적 쉬프트 연산자이다.

④ feedforward 은

a ^= aa; b -= bb; c + = cc;

dd ^= dd; e -= ee; f + = ff;

이다. 여기서 a, b, c, d, e, f레지스터는 192비트의 중간 해쉬 값인 h_i+ 1이며 알고리즘의 종료 후 최종 해쉬 값이 된다. 따라서 이 알고리즘의 출력인 32비트 SRES(Signed Response)와 암호키 64비트 는 다음 식에 의해 최종 생성된다.

SRES = a^b^c^d;

K_c = ef;

3.2.2 S-box

S-box는 비선형성높이기 위하여 x^{-1} 연산을 사용하였으며 일반적으로 역원을 구하는 연산은 많은 연산시간을 요구한다. 그러나 $GF(2^8)$ 에서의 역원이므로 미리 연산하여 참조 테이블을 구성하였다. 0의 역원은 존재하지 않으므로 0의 값으로 대응되는데 이는 암호학적으로 안전하지 않으므로 역원의 각 값에 0xa5값을 배타적-합(exclusive-OR)하여 256개의 8비트 값을 갖는 테이블을 구성하였다. S-box 테이블은 <표 1>에 나타내었다.

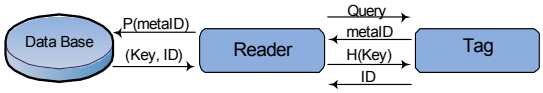
<표 1> S-box

0xa5	0xa4	0x33	0x41	0xee	0xf9	0xd7	0x60	0x16	0xe6	0x8b	0x58	0x9c	0x99	0x51	0x91
0x6a	0x52	0x12	0x97	0xb2	0x4c	0x4d	0xb1	0x2f	0x83	0xbb	0x7f	0xdf	0x15	0xbf	0x7e
0x54	0xec	0x48	0x68	0x68	0x1f	0xbc	0x2e	0x38	0x5b	0x47	0x5c	0xd1	0x08	0xaf	0x59
0xe0	0x44	0xb6	0x13	0xaa	0x50	0xc8	0x04	0x98	0xa9	0xfd	0x20	0xa8	0x9d	0x5e	0x19
0x4b	0x2a	0x17	0xac	0x45	0x95	0x06	0x56	0x55	0x84	0xf8	0xa1	0x3f	0x3d	0x76	0x0e
0x7d	0x6c	0xda	0xfa	0xd4	0x2c	0x4f	0xd8	0x9f	0x21	0x65	0xc3	0xa0	0xef	0xdb	0xf6
0xl1	0x6f	0x43	0x79	0x3a	0x67	0xfe	0x64	0x34	0xdc	0x49	0x86	0x05	0x93	0x63	0x28
0x2d	0xf1	0xa3	0x61	0x89	0x09	0x71	0x25	0x35	0xcc	0xb9	0x14	0x4e	0xf2	0xfb	0xf7
0xd2	0x70	0x74	0x7a	0xfc	0x9e	0x37	0x73	0xd5	0xf0	0xbd	0x82	0x62	0xca	0x4a	0xe4
0xdd	0xcd	0x23	0x72	0x1d	0x02	0xa7	0x40	0xe8	0x3e	0xe9	0x3c	0x5a	0x8d	0x66	0xc1
0xc9	0x92	0x57	0xe3	0x0c	0x0a	0x1c	0x30	0x0b	0x01	0x77	0xea	0xd0	0x88	0x0d	0x00
0xb8	0xde	0xe7	0xad	0xc5	0x6e	0x96	0xb7	0x31	0x03	0x80	0x69	0x9a	0x5f	0x1a	0x1b
0xff	0xc2	0xc0	0x3b	0xd6	0xa2	0xcb	0x29	0x7c	0xd4	0xc4	0x10	0x1e	0x81	0x53	0xb5
0x7b	0x27	0x0f	0xeb	0xd3	0x24	0x22	0x36	0xf5	0x6d	0xbe	0xba	0xc6	0x42	0x75	0x26
0xe1	0x94	0x8f	0x5d	0xa6	0x32	0xc7	0x78	0xb3	0xb0	0xf3	0xd9	0xcf	0x87	0xe5	0x2b
0xed	0x85	0x07	0xe2	0x 뀨	0x90	0x6b	0xb4	0x46	0x8e	0x18	0x9b	0x8a	0xae	0x8c	0x39

3.2.3 인증 설계

일방향 해시 함수의 역함수 계산 어려움에 기반한 Hash Lock에 PKI방법을 적용하여 MetaID를 비밀키로써 사용한다. Hash Lock 방식은 인가받지 않은 Reader기가 Tag를 읽는 것을 방지 할 수 있으며 탐지가능하며 Hash Function만을 요구하므로 저비용으로 구현가능하다. [그림 4]에서의 Reader는 미리 등록된 공개키(meta ID를 이용하여 생성된)로 Tag를 인증하고 meta ID로 각 Tag의 유일한 키(k)를 생성하며 이에 해당하는 meta ID = H(k)를 가지고 있다. 이 때 H()는 해쉬함수이다.

Tag는 자신의 비밀키를 이용하여 생성된 meta ID를 Reader에 보내고 Reader는 해당되는 키(k)를 만들어내고 Tag에 보낸다. 이때 Tag는 Reader로부터 보내어진 키(k)를 해쉬값과 자신의 meta ID를 비교하여, 그 값이 동일하면 자신의 ID를 전송한다.



[그림 4] 인증 프로토콜

(1) Reader는 Tag에게 질의를 보낸다

(2) Tag는 미리 생성된 비밀키를 이용한 생성된 Meta ID를 보낸다.

(3) Reader는 P(meta ID) 인증키를 생성한다. Reader는 Data Base에서 값을 조사하고 일치하면 Key와 ID를 Tag에게 전송 한다.

meta ID는 PKI와 관련하여 사용할 수 있는 장치들에 대하여 단일 접속이 가능하며 다중 요소 인증을 사용하여 지역 환경에서 접속가능 하다.

기 제안된 방법은 고정된 meta ID를 이용하여 공격할 수 있는 방법에 대하여 안전하고 인가 받지 않은 사용자를 접근을 방지하고 합법적인 Reader기에 의해서는 식별 가능하다. 공개키에 의해 암호화된 암호문의 주어진 Tag의 연결성을 감소시키기 위하여 주기적으로 재암호화를 수행함으로써 인증과 프라이버시가 안전하게 보장된다. 그리고, 해쉬함수와 공개키 알고리즘만을 사용함으로써 효율적이며 저비용으로 구현 가능하다.

4. 결론

본 논문에서 유비쿼터스 컴퓨팅 환경 내에서 발생할 수 있는 보안 취약점과 보안 요구 사항을 도출하여 제시하고, 인증기술을 중심으로 한 보안 서비스 방향을 제안함으로써 센서네트워크 및 RFID 기술의 현실화될 수 있는 기반기술인 보안 및 인증기술에 대하여 연구하였다. 유비쿼터스 환경에서 효율적이며 저비용 구현 가능한 Hash Lock과PKI를 사용한 인증 프로토콜을 설계하였다. 즉, 일방향 해시 함수의 역함수 계산 어려움에 기반한 Hash Lock기법과 PKI를 이용하여 인가받지 않은 사용자를 접근을 방지하고 합법적인 Reader기에 의해서는 식별 가능하도록 한다. PKI에 의해 암호화된 암호문의 주어진 Tag 연결성을 감소시키기 위하여 주기적으로 재암호화를 수행함으로써 인증과 프라이버시가 안전하게 보장된다. 그리고, 해쉬함수와 PKI만을 사용하여 효율적이고 저비용으로 구현 가능하다.

유비쿼터스 시대에는 사용자가 처한 시공간적 위치와 상황은 강력한 보안 체계를 필요로 하며 보안이 보장된 환경에서 사용자의 요구에 맞는 다양한 서비스들 제공 할 수 있다.

참고문헌

[1] A. J.Menezes, P. C. van Oorschot, S. A. Vanstone, Handbook of Applied Cryptography, CRC Press, 1997

[2] B. Preneel, "Analysis and design of cryptographic hash functions," Doctoral Dissertation, Katholieke Universiteit Leuven, 1993

[3] R. L. Rivest, "The MD4 message-digest algorithm," Advances in Cryptology-Crypto'90, Lecture Notes in Computer Science, Vol.537, pp.303-311, 1991

[4] R. L. Rivest, "The MD5 message-digest algorithm," Request For Comment(RFC) 1320, Internet Activities Board, Internet Privacy Task Force, April 1992

[5] FIPS 180-1, "Secure hash standard," Federal Information Processing Standards Publication 180-1, U.S. Department of Commerce/Nist, 1995

[6] H. Dobbertin, A. Bosselaers, B. Prennel, "RIPEMD160: A strengthened version of RIPEMD," Fast Software Encryption-Cambridge Workshop, Lecture Notes in Computer Science, Vol.1039, Springer-Verlag, pp.71-82, 1996

[7] Y. Zheng, J. Pieprzyk, J. Sebery, "Haval-a one-way hashing algorithm with variable length of output," Advances in Cryptology-AUSCRYPT '92, LNCS, Vol.718, pp.83-104, 1993

[8] Arati Manjeshwar et al, 'TEEN: A Routing Protocol for Enhanced Efficiency in Wireless Sensor Networks', Proc. Second Int'l Workshop', IEEE Proc. Of the Int'l. Parallel and Distributed Processing Symposium, 2002.