情報保護學會論文誌
第20卷 第1號, 2010. 2

개선된 수동형 RFID 보안태그와 리더의 인증 및 데이터 보호 프로토콜*

양 연 형,[1†] 김 선 영,[2] 이 필 중[1‡]
[1]포항공과대학교 전자전기공학과, [2]포항공과대학교 정보통신대학원

Improved Authentication and Data Protection Protocol of Passive RFID Security Tag and Reader*

Yeon Hyun Yang,[1†] Sun Young Kim,[2] Pil Joong Lee[1‡]
[1]Department of Electronic and Electrical Engineering, POSTECH,
[2]Graduate School of Information Technology, POSTECH

요 약

RFID는 사물에 전자 태그를 부착하고 무선통신 기술을 이용하여 사물의 정보 및 주변 상황 정보를 감지하는 인식 기술로써 다양한 영역에서 이용 범위가 확대되고 있으며 관련된 보안 문제 역시 크게 대두되고 있다. 현재까지 RFID 시스템의 안전성 문제를 해결하기 위해 많은 인증 프로토콜들이 제안되었다. 본 논문에서는 ISO 18000-6 Amd1[8]의 타입 C RFID 시스템과 호환되는 기존의 RFID 인증 프로토콜[9, 10]에 대하여 스푸핑 공격이 가능함을 보이고, 그 프로토콜을 개선하여 적은 양의 추가 비용을 갖는 개선된 프로토콜을 제안한다.

ABSTRACT

As an automatic identification technology, in which electronic tags are attached to items and system auto-identifies informations of the items using wireless communication technology, use of RFID system is increasing in various fields. According to that, related security problems are becoming important issue. Up to now, many authentication protocols have been proposed to solve security problem of RFID system. In this paper, We show that the RFID authentication protocols in [9, 10], which are compatible with Type C RFID system in ISO 18000-6 Amd1[8], are vulnerable to a spoofing attack. In addition, we propose improved protocols having small additional cost over the original protocols.

Keywords: RFID security, RFID authentication protocol, RFID standard

I. 서 론

최근 유비쿼터스 컴퓨팅[1]이 대두되면서 RFID/USN 등과 같은 기반 기술이 미래 사회의 중요한 요소가 될 것이라고 간주되고 있다. 특히 RFID(Radio Frequency Identification) 기술은 자동화된 물류 및 유통 분야에서 기존의 바코드가 하던 역할을 점차 대체해 가고 있다[2]. RFID 시스템을 사용하면 직접적인 접촉이 없이도 대상을 인식할 수 있으며, 리더가 한 번에 여러 개의 RFID 태그와 통신하는 것이 가능하여 전체 시스템의 효율을 획기적으로 개선할 수 있다. 그러나 대부분의 경우 RFID 태그는 단순한 연산만을 할 수 있고 통신거리도 길지 않은 등 제한이 많은 장치이며, 무선통신을 사용해야 한다.

접수일(2009년 1월 13일), 수정일(1차: 2009년 6월 15일, 2차: 2009년 9월 28일), 게재확정일(2010년 1월 19일)
* 본 연구는 Brain Korea 21 사업의 지원으로 수행하였습니다.
† 주저자, yhyang@oberon.postech.ac.kr
‡ 교신저자, pjl@postech.ac.kr

무선통신을 태그 인식에 사용하는 RFID 시스템에서 보안은 중요한 선결과제가 된다. 무선통신은 그 특성상 도청, 변조와 같은 위협에 노출되어 있으며[3], RFID 태그에 담기는 정보들은 디지털 정보이므로 태그를 불법적으로 복제하는 것도 가능하다. 따라서 안전한 RFID 시스템에서는 태그가 정당한 것이었는지를 인증하는 것이 매우 중요한 문제이다. 그리고 태그가 리더에게 전송하는 정보 또한 개인의 프라이버시(privacy)를 침해하는 데에 이용될 수 있으므로 태그 자신이 정당한 리더와 통신하고 있는지를 확인하는 것 또한 중요하다[4].

리더와 태그 사이의 인증을 위해서 초기에 제안되었던 대표적인 프로토콜에는 해시-락(hash-lock)을 이용하는 기법[5]이 있으며, 이 기법에서는 태그의 실제 ID를 숨기기 위해 metaID를 사용하였다. 그 외에 인증 방법에는 해시 함수의 특징과 도전-응답(challenge-response) 방법을 결합한 기법[6], 동기화된 비밀 값에 기반하는 기법[7] 등이 있다. 이러한 기존 연구들은 태그의 인증이나 리더와 태그의 인증 외에 위치 추적 방지나 프라이버시 보호 등을 제공하는 것을 목적으로 하며 그 과정에서 자연스럽게 태그의 ID, 또는 그와 관련한 정보들이 프로토콜 실행 시마다 갱신되는 특성이 있다. 그러나 본 논문에서 다루고 있는 RFID 인증 프로토콜은 ISO 18000-6 등의 국제표준[8]과 호환되는 RFID 시스템을 위한 것으로 태그는 고정된 ID를 갖는 특징이 있다.

RFID 시스템에서의 태그 인증, 또는 리더-태그 상호 인증 등은 국제적으로도 많은 관심 속에 연구되고 있으며 ISO 18000-6과 같은 국제 표준화 작업도 활발히 이루어지고 있다[8]. 국내에서도 ISO 18000-6의 타입 C RFID 시스템과 호환되는 RFID 인증 프로토콜이 제안되었으며[10], 이와 관련한 국내표준으로는 2008년 12월에 한국정보통신기술협회(TTA)에서 제정한 잠정표준[1](TTAI.KO-12.0091)인 "수동형 RFID 보안태그와 리더의 인증 및 데이터 보호 프로토콜[9]"이 있다. 이 잠정표준에 기술된 프로토콜들은 저가형 RFID 시스템에서 AES 알고리즘을 이용한 리더와 태그 간의 상호인증, 태그의 단방향 인증을 수행하는 프로토콜들이다.

그런데 해당 잠정표준에 기술된 인증 프로토콜에는 보안상 개선할 점이 존재한다. 해당 표준에 기술되어 있기로는 공격자가 세션키 생성에 사용되는 Key_RN을 리더에게 재전송할 경우 둘 이상의 세션에서 동일한 세션키가 사용되는 잠재적인 위험이 있다고만 하였으며, 또한 Key_RN을 재전송하는 가짜 태그가 있다고 해도 주어진 응답제한시간 안에 인증을 성공할 수 없을 것이라고 기술하고 있다. 그러나 본 논문에서는 Key_RN을 재전송하는 공격자가 있을 경우 가능한 효율적이고 명시적인 공격 방법을 제시하여 이것이 잠재적인 위험이 아닌 현실적인 스푸핑이 가능함을 보이고자 한다. 또한 이 문제의 원인 분석과 개선책을 제시하며 적은 양의 추가 비용을 갖는 개선된 프로토콜들을 제안한다.

II. 인증 프로토콜

먼저 [9, 10]에서 기술하고 있는 인증 프로토콜에 대하여 기술하도록 한다. 이후에 설명하는 프로토콜들은 보안 기능을 사용하지 않을 경우 ISO 18000-6 Amd1[8]에서 정의된 타입 C RFID 시스템과 호환되도록 설계되었으며, 보안 기능을 사용할 경우 AES를 이용한 추가적인 인증작업을 수행하도록 되어 있다. 이후에서는 AES를 사용하여 인증을 수행하는 프로토콜들만 다루도록 한다.

다음은 앞으로 설명할 프로토콜에서 사용될 주요 약어들이다.

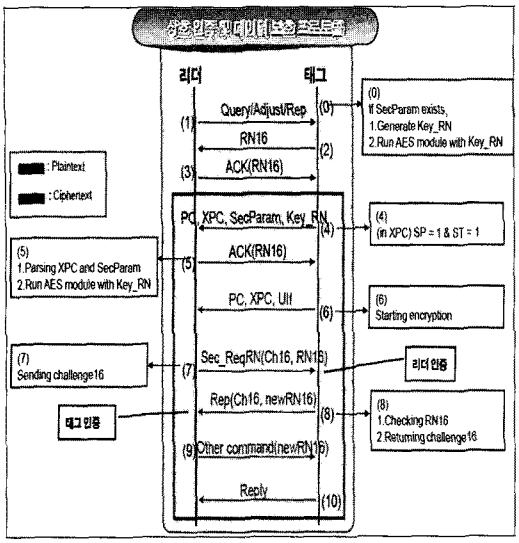
- PC: Protocol Control
- XPC: eXtended Protocol Control
- UII: Unique Item Identification
- RN: Random Number
- SecParam: Security Parameter

위에서 PC, XPC 등은 보안 메커니즘을 적용할 것인지의 여부나 프로토콜을 어떠한 방식으로 수행할 것인지 등의 제어 정보를 나타내는 값으로 태그의 신원정보와는 직접적인 관련이 없다. UII는 태그 고유의 신원정보를 나타내는 값으로 프로토콜 수행 이후에도 계속 고정된다[8]. 또한 SecParam은 태그의 보안기능 지원여부, AES 라운드 수[2], 키 인덱스 등 보안에 관련된 파라미터들의 정보를 가지고 있는 값이다[9].

1) 잠정표준이란 표준을 조속히 제정할 필요가 있으나 기술발전추세 등의 확인에 오랜 시간이 걸리는 경우 일시적으로 적용되는 표준을 말한다. 또한, 제정 후 1년 이내에 국문표준 또는 영문표준으로의 채택(개정작업)을 심의(추진)하여야 한다.

2) AES 라운드 수를 지정하는 이유는 데이터 처리 능력이 떨어지는 태그의 경우에 AES 라운드 수를 줄여서 리더가 요구하는 응답시간 안에 쿼리 응답을 전송할 가능성을 주기 위함이다.

1. 리더-태그 상호인증 및 데이터 보호 프로토콜



(그림 1) 상호 인증 및 데이터 보호 프로토콜

이 프로토콜은 사전에 키를 공유한 리더와 태그 사이의 상호인증을 다루고 있으며[8, 9], 따로 백-엔드(back-end) 쪽의 데이터베이스 서버 등을 가정하고 있지는 않다. 따라서 상호인증은 리더와 태그가 서로 올바른 키를 공유하고 있는지를 확인하는 과정이라고 할 수 있다. 태그와 리더는 미리 정의된 키 풀(pool)[3]을 가지고 있으며, 인증 과정에서는 어떤 키를 사용할 것인지 태그가 리더에게 해당 키의 인덱스를 SecParam에 포함하여 전달한다. 이 키를 공유키 K라 하도록 한다. 프로토콜 절차는 (그림 1)과 같으며, (4)~(10)단계를 제외한 부분은 18000-6 Amd1[8]의 과정과 동일하다.[4]

그림에서 Query/Adjust/Rep, ACK, Sec_Req RN 등은 RN16과 같은 변수명이 아닌 명령어 코드를 의미한다. Query/Adjust/Rep는 Query, Adjust와 Rep의 세가지 명령어 코드 중 하나를 보낸다는 의미인데 이 명령어 코드들은 모두 18000-6 Amd 1에서 정의하는 인벤토리 과정(Inventory Operation)을 시작하는 명령들이다[8]. 단계 (6)~(9)에서 사용되는 암호화는 스트림 암호와 유사한 방식으로 동작한다. 이후 본문 중의 설명에서 Enc(A)는 A를 암호화한 값을 의미한다. 자세한 암호화 방법은 II.3에서 설명한다.

(2) 단계에서 리더가 태그에게 전송하는 RN16은 태그가 생성하는 16-비트 난수이며 handle이라 불린다. 이 handle은 여러 개의 태그를 상대하는 리더에게 태그가 자신을 다른 태그로부터 구분하기 위한 임시 ID로 볼 수 있다[5]. 또한 여러 개의 태그 사이에 섞여 있는 특정 태그가 자신에게 전달되는 리더의 명령을 구분해 내기 위해서도 사용된다. (7) 단계에서와 같이 handle이 암호문 속에 포함된 경우는 태그가 암호문을 풀어서 해당하는 handle이 복호화 되는지를 확인해야 한다.

(4) 단계에서 태그가 16-비트 Key_RN을 임의로 생성하여 리더에게 전송한 이후 태그와 리더가 모두 Key_RN을 알게 되고 K와 Key_RN으로부터 공통된 세션키를 생성한다. 이 세션키는 이후 (6)~(9) 단계의 통신에서 암호문을 생성하는 데 사용된다. 리더가 (7) 단계에서 RN16이 포함된 암호문을 태그에게 보내게 되는데, 태그는 암호문을 복호화 하여 자신이 보냈던 RN16이 포함되어 있는지를 확인함으로써 리더가 제대로 된 K를 공유하고 있었음을 확인한다[리더 인증]. Ch16은 리더가 태그를 인증하기 위한 챌린지로 사용되는 16-비트 난수이며 리더가 생성한다. 태그가 (8) 단계에서 Ch16이 포함된 암호문을 리더에게 보내게 되는데, 리더는 암호문을 복호화하여 자신이 보냈던 Ch16이 포함되어 있는지를 확인함으로써 태그가 제대로 된 K를 공유하고 있었음을 확인한다[태그 인증]. 이 단계가 성공하면 (6) 단계에서 태그가 전송한 암호화된 UII가 정당한 것이었음을 확인할 수 있다. newRN16은 (9) 단계, 혹은 그 이후에 리더가 태그에게 추가적인 명령을 전송할 경우 태그가 리더를 인증하기 위해서 사용된다.

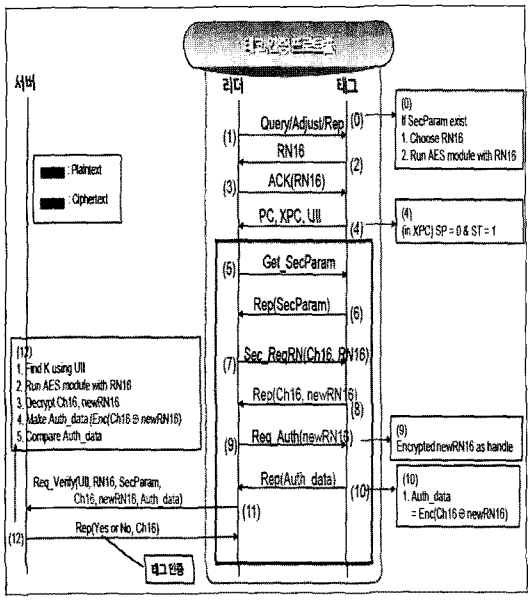
2. 태그 인증 프로토콜

이 프로토콜에서는 리더와 태그가 키를 공유하지 않는다. 따라서 태그가 정당한 것이었는지를 인증하기 위해서는 태그와 키를 공유하고 있는 제3자가 필요하다. 이 역할은 백-엔드 데이터베이스(back-end database)가 맡도록 기술되어 있다. 단, 태그에게

3) 실제 태그가 가지고 있는 키 풀은 리더가 가지고 있는 키 풀 중에 일부분만을 갖는다.

4) ISO 18000-6 Amd1의 인증과정의 경우 단계 (4)에서는 태그가 PC, XPC, UII를 리더에게 전송하고 인증은 종료된다.

5) 여러 개의 태그 사이에 섞여 있는 하나의 태그를 골라내는 인벤토리(Inventory) 과정의 자세한 사항은 ISO 18000-6 Amd1[8]에 기술되어 있다.



(그림 2) 태그 인증 프로토콜

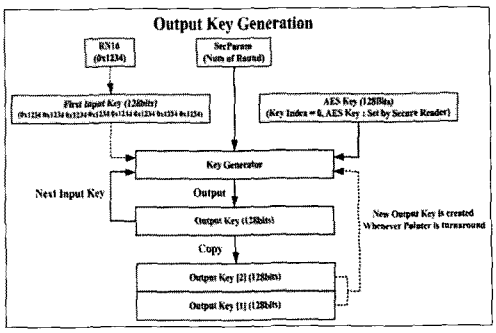
16-비트 랜덤 챌린지를 보내는 역할은 리더가 맡는다. 태그 인증 프로토콜 절차는 (그림 2)와 같다. (4)까지의 단계는 ISO 18000-6 Amd1[8]의 과정과 동일하다.

유의할 것은 (9)단계에서 리더가 태그에게 보내는 Enc(newRN16)은 리더가 새로 생성하는 암호문이 아니고 (8)단계에서 태그가 보냈던 Enc(newRN16)을 그대로 전송한다는 것이다. 이것은 암호화된 값 자체를 의미하며 이전의 RN16 대신에 태그의 handle 역할을 한다. 태그 인증 프로토콜에서는 리더-태그 상호 인증 프로토콜에서와는 다르게 태그가 Key_RN을 별도로 생성하지는 않고, 자신의 임시 ID로 사용되는 RN16과 마스터 키 K로부터 세션키를 생성한다. 즉, 세션키 생성과정의 입력이 Key_RN과 K가 아니라 RN16과 K가 된다.

(10)~(12) 단계에서 태그 인증을 실제로 수행하는 데 필요한 정보인 Auth_Data는 Enc(Ch16⊕newRN16)이다. 데이터베이스는 이 Ch16과 new-RN16로부터 Auth_Data가 제대로 생성된 것인지 확인하여 태그를 인증하게 된다.

3. 세션키 생성 방법 및 암호화 방법

(그림 3)은 세션키를 생성하고 사용하기 위한 전체 구조를 나타내며, 실제 세션키의 생성은 키 생성기

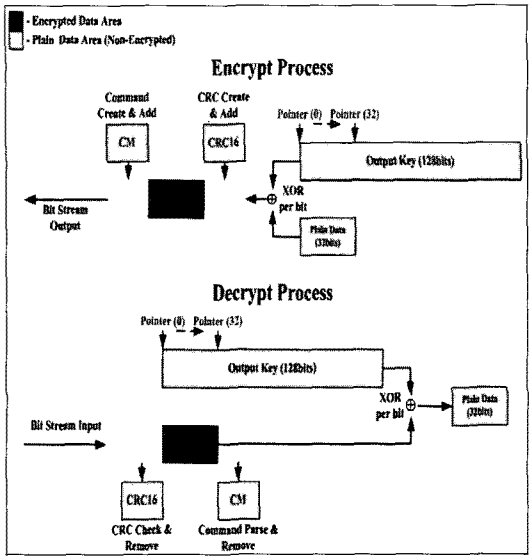


(그림 3) 세션키 생성 방법

(Key Generator)가 수행한다. 세션키는 128-비트 AES 암호 알고리즘을 이용하여 생성하며 블록암호의 동작 모드 중에서 OFB(Output Feedback)와 같은 방식으로 동작한다. OFB 모드의 IV(Initial Value)로는 Key_RN을 사용하고 AES 키로는 마스터키 K를 사용한다. 첫번째 AES 실행에서는 IV를 평문 입력으로 사용하고, 이후 세션키를 더 생성하기 위해 AES 알고리즘을 추가로 실행하는 경우에는 직전에 생성된 AES 출력을 평문 입력으로 feedback 한다. AES 암호 알고리즘의 출력을 N 비트라 하고 (6)~(7) 단계에서 전달해야 하는 평문의 총 길이가 M 비트일 때 모두 $\lceil M/N \rceil$ 번의 AES 알고리즘을 수행해야 한다. 이 세션키는 스트림 암호의 키 스트림과 같은 역할을 한다. 키 생성기의 동작 방식을 알고리즘으로 나타내면 다음과 같다.

Input: K, IV
1. M ← IV
2. C ← AES(K, M)
3. Output C and sleep until invoked
4. M ← C
5. Goto step 2

(그림 4)는 암호화 및 복호화 방법을 나타낸다. 앞에서 생성된 세션키(키 스트림)는 암호화하고자 하는 평문과 비트 단위로 XOR 된다. 실제로는 32 비트 단위로 동작하며 처음에는 포인터를 세션키의 첫번째 비트에 둔다. 입력되는 평문 32 비트와 세션키의 포인터 위치로부터 32 비트를 추출하여 비트 단위로 XOR하여 암호문을 생성한다. 포인터는 32 비트만큼 이동한다. 평문을 32 비트씩 처리하면서 포인터가 AES 출력 128 비트를 다 지나고 나면 앞에서 기술한 키 생성기를 호출하고 AES 알고리즘을 한 번 더 수행하여 다시 128 비트를 생성한다. 이 때 포인터를 다시 128 비트 중 첫 비트로 옮긴다. M 비트의 평문을 암호화



〔그림 4〕 암호화 및 복호화 방법

하기 위해서는 $\lceil M/N \rceil \cdot N$ 비트의 세션키가 필요하다. 복호화를 위해서는 역시 동일한 방식으로 세션키를 생성한 후 이것을 암호문과 비트 단위로 XOR하여 평문을 얻는다.

III. RN16과 Key_RN의 재전송을 이용한 스푸핑 공격

여기서는 소단원에 관한 내용을 간단히 살펴보겠습니다. 계다가 소소단원에 관한 내용도 간단히 살펴보겠습니다.

1. 리더-태그 상호인증 프로토콜 공격

공격자는 사전에 특정 태그의 인증과정을 기록해 놓았다고 가정한다. 이 기록된 인증과정에서 사용된 세션키와 리더가 전송한 랜덤 챌린지를 각각 SK^{prev}와 $Ch16^{\text{prev}}$라 하고, 공격자가 가짜 인증을 수행할 때 사용되는 세션키와 챌린지를 각각 SK^{curr}와 $Ch16^{\text{curr}}$라 한다.

공격자는 다음 과정을 통해서 가짜 인증을 수행한다.

- 공격자는 태그를 가장하여 (1)~(6) 단계에서는 이전 인증과정의 내용을 재전송한다.

재전송의 결과 같은 Key_RN이 리더에게 전달되므로 이전 인증과정과 같은 세션키가 사용된다. 즉, $SK^{\text{prev}} = SK_1 \| SK_2 \| \cdots \| SK_n$이었다면 현재 (가짜) 인증과정의 세션키도 역시 $SK^{\text{curr}} = SK_1 \| SK_2 \| \cdots \| SK_n$가 된다. 이 때 $n = \lceil M/32 \rceil$이다. 결국 같은 단계에서 전달되는 암호문에는 이전 인증과정과 같은 세션키 블록 SK_i가 사용된다. (6) 단계에서 태그가 전달하는 암호문은 $(SK_1 \oplus \text{PC}) \| (SK_2 \oplus \text{XPC}) \| (SK_3 \oplus \text{UII})$의 형태가 된다. 실제 프로토콜에서 SK_1과 SK_2는 각각 PC와 XPC와 같은 길이여야 한다. 그러나 공격에 영향을 미치는 값은 SK_1, SK_2나 PC, XPC 등의 값이나 길이가 아니고 $SK_3 \oplus \text{UII}$이 매번 같은 비트 위치에 있는지 여부이기 때문에 기술의 편의상 SK_i와 PC, XPC, UII는 모두 16-비트로 같은 크기라고 가정한다.

- (7) 단계에서 공격자는 리더로부터 암호문 $(SK_4 \oplus Ch16^{\text{curr}}) \| (SK_5 \oplus RN16)$을 전송받는다.

이전 인증과정의 (7) 단계에서 태그가 받은 암호문을 $(SK_4 \oplus Ch16^{\text{prev}}) \| (SK_5 \oplus RN16)$라 하고, 현재 인증과정에서 공격자가 새로 받은 암호문을 $(SK_4 \oplus Ch16^{\text{curr}}) \| (SK_5 \oplus RN16)$라 한다. 이로부터 다음의 계산을 이용하여 $Ch16$의 차분 $\Delta Ch16$을 생성한다.

$$\Delta Ch16 = (SK_4 \oplus Ch16^{\text{prev}}) \oplus (SK_4 \oplus Ch16^{\text{curr}}) = Ch16^{\text{prev}} \oplus Ch16^{\text{curr}}$$

이전 인증과정의 (8) 단계에서 태그가 리더에게 전달한 암호문을 $(SK_6 \oplus Ch16^{\text{prev}}) \| (SK_7 \oplus \neq wRN16^{\text{prev}})$라 하면 현재 인증과정에서 공격자가 전달해야 하는 암호문의 $SK_6 \oplus Ch16^{\text{curr}}$ 부분은 다음 계산을 통해서 얻는다.

$$SK_6 \oplus Ch16^{\text{curr}} = SK_6 \oplus Ch16^{\text{prev}} \oplus \Delta Ch16 = SK_6 \oplus Ch16^{\text{prev}} \oplus Ch16^{\text{prev}} \oplus Ch16^{\text{curr}}$$

- (8) 단계에서 공격자는 $(SK_6 \oplus Ch16^{\text{curr}}) \| (SK_7 \oplus \neq wRN16^{\text{curr}})$을 전송한다.

여기에서 $\neq wRN16^{\text{curr}}$는 태그, 즉 공격자가 선택하는 것이기 때문에 실제로는 $SK_7 \oplus \neq wRN16^{\text{curr}}$ 대신에 같은 길이의 랜덤 비트열로 대체해도 상관없다. 즉, 세션키 블록인 SK_7을 정확히 알아야 할 필요는 없다. 리더는 $(SK_6 \oplus Ch16^{\text{curr}})$ 부분을 복호화하여 $Ch16^{\text{curr}}$가 나오는지를 확인하게 되므로 공격자는 가짜 인증과정을 성공적으로 수행할 수 있다.

위 설명에서는 (6) 단계에서 이전 인증과정의 암호

문 $(SK_1 \oplus \text{PC}) \| (SK_2 \oplus \text{XPC}) \| (SK_3 \oplus \text{UII})$을 재전송하는 것으로 기술하였는데, 이럴 경우 이전 인증과정에서 사용된 UII에 대한 인증을 수행하는 것이 된다. 그런데 공격자가 이 UII를 사전에 알고 있었다면 임의의 UII′를 대신 삽입하여 인증을 수행할 수 있다. 이전 인증과정의 (6) 단계에서 전달된 암호문 중 UII 부분을 $SK_3 \oplus \text{UII}$라 하면 공격자는 임의로 선택한 UII′에 대해서 $SK_3 \oplus \text{UII} \oplus \text{UII} \oplus \text{UII}' = SK_3 \oplus \text{UII}'$의 계산을 통해 UII′의 올바른 암호문을 만들 수 있다. 또한 이전 인증과정에서 사용한 UII를 정확하게 모르더라도 UII와 연관된 UII′의 암호문을 만들 수 있다. 예를 들면 임의의 위치의 비트들을 뒤집은 새로운 UII′에 대한 암호문을 만드는 것 등이다.

2. 태그 인증 프로토콜 공격

태그 인증 프로토콜에서도 세션키의 신선도는 태그에 의해서만 결정되므로 리더-태그 상호 인증 프로토콜에서와 같은 문제가 발생한다. 즉, 공격자는 이전 인증과정의 통신 내용 중 전부 또는 일부(키 인덱스와 RN16 등)를 재전송함으로써 이전에 사용됐던 세션키가 재사용되도록 상황을 조절할 수 있다. 공격 과정은 다음과 같다.

- 공격자는 태그를 가장하여 (1)~(6) 단계에서는 이전 인증과정의 내용을 재전송한다.
- (7) 단계에서 공격자는 리더로부터 명령어 코드와 $Ch16^{curr} \| RN16^{curr}$을 전송받는다.

(8) 단계에서 태그가 리더에게 보내야 하는 암호문은 $(SK_1 \oplus Ch16^{curr}) \| (SK_2 \oplus \neq wRN16^{curr})$이다. 이전 인증과정의 (7) 단계에서 리더가 태그에게 전송했던 챌린지를 $Ch16^{prev}$라 하고 태그가 리더에게 전송했던 암호문을 $(SK_1 \oplus Ch16^{prev}) \| (SK_2 \oplus \neq wRN16^{prev})$라 할 때 공격자는 다음 계산을 한다. 아래의 수식에서 Δnew는 공격자가 임의로 선택하는 값이다.

$$\Delta Ch16 = Ch16^{prev} \oplus Ch16^{curr}$$
$$SK_1 \oplus Ch16^{curr} = (SK_1 \oplus Ch16^{prev}) \oplus \Delta Ch16$$
$$SK_2 \oplus \neq wRN16^{curr} = (SK_2 \oplus \neq wRN16^{prev}) \oplus \Delta \neq w$$

- (8) 단계에서 공격자는 리더에게 $(SK_1 \oplus Ch16^{curr}) \| (SK_2 \oplus \neq wRN16^{curr})$을 전송한다.
- (9) 단계에서 공격자는 명령어 코드와 함께 $(SK_2 \oplus \neq wRN16^{curr})$을 전송받는다.

II.2 절에서 설명했듯이 리더는 (8) 단계에서 공격자로부터 전송받은 $SK_2 \oplus \neq wRN16^{curr}$을 그대로 전송하게 된다.

(10) 단계에서 공격자가 리더에게 보내야 하는 암호문은 $(SK_3 \oplus Ch16^{curr} \oplus \neq wRN16^{curr})$이다. 이전 인증과정의 (10) 단계에서 리더가 태그에게 전송했던 암호문을 $(SK_3 \oplus Ch16^{prev} \oplus \neq wRN16^{prev})$이라 할 때 공격자는 다음의 계산을 한다. 아래의 수식에서 $\Delta \neq w$는 공격자가 앞서 선택했던 값이다.

$$SK_3 \oplus Ch16^{curr} \oplus \neq wCh16^{curr} = (SK_3 \oplus Ch16^{prev} \oplus \neq wCh16^{prev}) \oplus \Delta Ch16 \oplus \Delta \neq w$$

- (10) 단계에서 공격자는 리더에게 $(SK_3 \oplus Ch16^{curr} \oplus \neq wRN16^{curr})$를 전송한다.

이후 (11) 단계에서 리더가 공격자에게서 전송받은 암호문을 백-엔드 데이터베이스에 보내면 데이터베이스에서는 공유키로부터 만들어지는 SK_3와 리더로부터 전송받은 정보로부터 Auth_data인 $(SK_3 \oplus Ch16^{curr} \oplus \neq wRN16^{curr})$을 생성해서 비교하게 되므로 인증이 성공한다.

이 공격이 가능한 이유는 리더가 태그에게 보내는 챌린지가 평문으로 전달되기 때문이고, 태그(공격자)가 선택해야 하는 $\neq wRN16^{curr}$는 임의의 $\Delta \neq w$를 고른 후 $\neq wRN16^{prev} \oplus \Delta \neq w$의 계산을 통해서 얻을 수 있기 때문이다. 실제 정상적인 프로토콜에서도 $\neq wRN16^{curr}$는 태그가 임의로 선택하는 값이기 때문에 리더의 입장에서는 차이가 없다. $\Delta \neq w$를 0^{16}으로 정하여 이전에 사용됐던 $\neq wRN16$을 재전송할 수도 있다.

표준에서는 향상된 태그 인증 프로토콜이라 하여 효율을 향상시킨 또 하나의 태그 인증 프로토콜을 기술하고 있는데, 이 향상된 프로토콜에서 리더가 태그에게 보내는 챌린지가 $Ch16$일 때 태그가 생성해 내야 하는 정보는$(SK_1 \oplus \neq wRN16) \| (SK_2 \oplus Ch16 \oplus \neq wRN16)$가 된다. 이 경우에도 공격자는 $\Delta Ch16$과 $\Delta \neq w$를 통해서 마찬가지로 원하는 정보를 생성할 수 있다.

3. 원인 및 개선책

앞서와 같은 공격이 가능한 이유는 두 가지이다. 우선 암호화 과정에서 사용되는 세션키를 만들 때 마스터키 K와 태그가 생성한 Key_RN(태그 인증 프로토

콜에서는 RN16)만 관여한다는 것이다. 리더는 세션키의 신선도(freshness)에 기여하는 바가 없으므로 실제로는 태그가 세션키를 생성하여 리더(태그 인증 프로토콜에서는 데이터베이스)에게 전달한 것과 같은 효과를 내게 된다.

취약점의 두 번째 원인은 암호화 과정이 세션키와 평문의 단순한 XOR라는 것이다. 여기에서 세션키는 스트림 암호의 키 스트림과 같은 역할을 하게 된다. 이러한 방식에서는 오류(error)가 전파(propagation)되지 않기 때문에 키 스트림이 일단 정해지고 나면 암호문의 각 비트들은 독립적으로 처리된다. 따라서 리더, 혹은 태그는 암호문 중 특정 내용만을 위조할 수가 있다.

이것을 형식적으로 나타내면 다음과 같다. 임의로 선택된 두 개의 키 스트림 블록 SK_1과 SK_2에 대하여, 평문 M_1과 M_2의 각 키 블록에 대한 암호문을 $C_{11}=SK_1\oplus M_1$, $C_{12}=SK_1\oplus M_2$, $C_{21}=SK_2\oplus M_1$과 $C_{22}=SK_2\oplus M_2$라 하자. 이 때 공격자에게 4개의 암호문 중 C_{11}, C_{12}, C_{21}가 주어지면 공격자는 C_{22}를 $C_{22}=SK_2\oplus M_2=C_{11}\oplus C_{12}\oplus C_{21}$을 통해서 만들 수 있다. 이를 변형하여 C_{11}과 C_{12}는 이전과 같고 $C_{21}=SK_2\oplus M_1\oplus D$일 때 $C_{22}=SK_2\oplus M_2\oplus D$를 구하는 것도 마찬가지로 쉽게 풀 수가 있다. 이를 방지하기 위해서는 C_{11}과 C_{12}는 이전과 같고 $C_{21}=SK_2\oplus M_1\oplus D_1$일 때 $C_{22}=SK_2\oplus M_2\oplus D_2$를 푸는 문제로 변형하고 공격자가 D_1과 D_2의 관계를 알 수 없도록 해야 한다. 결국 이 문제는 조건이 앞과 같은 때 또 다른 키 스트림 블록 SK_2'에 대하여 $C_{22}=SK_2'\oplus M_2\oplus D_1$를 구하는 문제로 귀결된다. 이를 앞서의 프로토콜에 적용하면 이전에 사용됐던 세션키(키 스트림)가 다시 사용되지 않도록 해야 한다는 의미가 된다.

IV. 개선된 프로토콜

1. 개선된 리더-태그 상호 인증 프로토콜

본 절에서는 II.1 프로토콜에서 바뀐 부분만을 설명할 것이다. 이전 세션키가 재사용되지 않도록 하기 위해서는 리더와 태그가 각각 랜덤 비트열을 만들어 이로부터 세션키를 생성해야 한다. 이 각각의 랜덤 비트열을 Key_RN1과 Key_RN2라 하면 리더-태그 상호 인증 프로토콜의 (4),(5)단계는 아래와 같이 바뀌어야 한다.

(4) 리더 ← 태그: PC, XPC, SecParam, **Key_RN1**

(5) 리더 → 태그: **"ACK_Sec", Key_RN2,** RN16

다른 단계는 원 프로토콜과 모두 같으며 (5) 단계에서 {"ACK", RN16}을 전송하던 것을 새로운 명령어 "ACK_Sec"을 이용하여 {"ACK_Sec", Key_RN2, RN16}을 전송하도록 한다. (4)~(5) 단계를 통하여 리더와 태그가 Key_RN1과 Key_RN2를 교환하고 이들 랜덤 비트열과 공유키 K를 이용하여 실제의 세션키를 생성한다.

이러한 방식을 사용하기 위해서는 "ACK_Sec"라는 새로운 명령어를 ISO 18000-6 Amd1의 인벤토리 과정(Inventory operation)의 명령어 모음에 포함시켜야 한다. 이 명령어는 ISO 18000-6 Amd1에서 정하고 있는 custom command로는 지정할 수가 없는데, 그 이유는 리더가 custom command를 전송하기 위해서는 일단 그 전에 태그의 신원을 알아내도록 기술하고 있기 때문이다.

2. 개선된 태그 인증 프로토콜

본 절에서도 마찬가지로 II.2 프로토콜에서 바뀐 부분만을 설명할 것이다. 기본적인 수정 원리는 상호 인증 프로토콜과 같다. 즉, 리더가 랜덤 비트열을 생성하여 태그에게 전달하고 태그는 자신이 선택한 랜덤 비트열(RN16)과 리더로부터의 랜덤 비트열(Key_RN2)을 모두 이용하여 세션키를 생성한다. 따라서 태그 인증 프로토콜의 (5)단계는 아래와 같이 변경된다.

(5) 리더 → 태그: "Get_SecParam", **Key_RN2, RN16**

나머지 과정은 원래의 프로토콜과 같으며 (5) 단계에서 리더가 태그에게 {"Get_SecParam", RN16}을 전송하던 것을 {"Get_SecParam", Key_RN2, RN16}을 전송하도록 하는 것이다. 이를 위해서는 "Get_SecParam" 명령어의 파라미터로 handle인 RN16만을 보내던 것을 16-비트 랜덤 비트열인 Key_RN2도 같이 전달하도록 명령어 형식을 수정하여야 한다.

3. 수정된 세션키 생성 방법

위의 두 가지의 개선된 프로토콜에서는 세션키를 생성하기 위해서 두 개의 랜덤 비트열과 마스터 키를

사용해야 한다. 연속되는 AES 알고리즘의 IV (Initial Value)로 Key_RN1 ‖ Key_RN2, 혹은 RN16 ‖ Key_RN2를 사용하도록 수정한다. 나머지 과정은 원래의 세션키 생성 방법과 동일하며, 알고리즘으로 표현하면 아래와 같다.

Input: K, IV1, IV2
1. M ← IV1 ‖ IV2
2. C ← AES(K, M)
3. Output C and sleep until invoked
4. M ← C
5. Goto step 2

V. 안전성 분석

이 절에서는 RFID 인증 프로토콜에 적용되는 주요 공격에 대하여 IV 절의 프로토콜이 안전함을 보인다. 도청 공격과 위치 추적에 대한 안전도는 기존 프로토콜과 차이가 없으므로 여기에서는 기존 프로토콜에 난수 하나가 더 추가됨으로 인해서 어떤 안전도의 개선이 있는가 하는 것을 중심으로 설명할 것이다.

1. 재생 공격(Replay attack)

II 절의 기존 프로토콜들은 이전에 기록된 프로토콜의 통신 내용 중 세션키를 생성하기 위한 정보들을 재전송함으로써 스푸핑이 가능하였다. 그러나 IV 절의 개선된 프로토콜에서는 리더와 태그가 모두 세션키 생성을 위한 IV(initial value)를 고르게 된다. II 절의 프로토콜에서는 III 절에서 보았듯이 공격자가 Key_RN을 재전송할 경우 확률 1로 이전에 쓰였던 것과 같은 세션키를 사용하도록 조절할 수 있었다. IV 절의 프로토콜에서는 공격자가 Key_RN1을 재전송하더라도 리더가 선택하는 16-비트 랜덤 비트열인 Key_RN2 역시 재사용되지 않으면 이전에 쓰였던 것과 같은 세션키가 생성되지 않는다. 즉, 같은 세션키가 재사용될 확률은 Key_RN2가 우연히 재사용되는가에 좌우되고 그 확률은 2^{-16}이다. 이 확률은 통상적으로 고려되는 암호학적 안전도(예를 들어 2^{-80})와 비교하면 높은 확률이다. 그러나 표준에서 제시하고 있는 인증 프로토콜의 전체적인 안전도가 2^{-16}의 확률에 근거하고 있으므로 개선된 프로토콜에서도 최대한으로 얻을 수 있는 안전도는 이것이 한계가 된다.

2. 중간자 공격(Man-in-the-middle attack)

중간자 공격은 현재 통신하고 있는 상대의 신원을 완전히 믿을 수 없는 경우 발생한다. 그러나 본 논문에서와 같이 사전에 공유된 비밀키를 사용하는 인증의 경우에는 이와 같은 공격이 가능하지 않다. 공격자가 자신의 정보를 끼워 넣기 위해서는 양쪽의 통신 당사자들이 정당한 비밀키 대신 공격자의 키를 사용하도록 해야 하는데 이것이 불가능하기 때문이다.

하지만 완전한 형태의 중간자 공격은 아니라 해도 공격자가 태그의 UII를 미리 알고 있을 때 스트림 암호의 근본적인 문제로 인한 공격은 가능하다. 이 공격은 기존의 프로토콜과 개선된 프로토콜 모두에 해당한다. 예를 들어, II.1 절의 프로토콜의 경우 (6) 단계에서 태그가 전송하는 정보 중에 UII의 암호문이 포함되는데, $SK_i \oplus UII$의 형태의 암호문에서 $SK_i \oplus UII \oplus UII \oplus UII'$의 계산을 통해서 공격자가 선택한 임의의 UII'에 대한 암호문으로 치환 가능하다. 일단 UII가 노출된 상황에서는 이러한 공격을 막을 수가 없으므로 UII를 철저히 보호하는 것이 필요하다.

VI. 결론

본 논문에서는 [9, 10]에 기술된 RFID 인증 프로토콜에 대하여 스푸핑 공격이 가능함을 보이고, 원인 분석과 개선책을 제시하였다. 공격자가 Key_RN을 재전송하여 이전에 쓰였던 세션키를 재사용하게 함으로써 스푸핑 공격을 수행할 수 있었으며 이는 리더가 세션키의 신선도(freshness)에 기여하는 바가 없기 때문이었다. 본 논문에서는 이러한 취약점을 개선하기 위해 리더도 랜덤 비트열을 생성하여 태그에게 전달하고 리더의 랜덤 비트열과 태그의 Key_RN을 모두 사용하여 세션키를 생성하도록 하는 개선된 프로토콜을 제시하였다. 또한 이를 위하여 18000-6 Amd1과 호환되는 명령어 형태를 제안하였다. 이렇게 수정된 프로토콜은 기존 프로토콜에 적은 비용만을 추가하여 얻을 수 있는 것이다.

참 고 문 헌

[1] M. Weiser, "Some Computer Science Issues in Ubiquitous Computing," Communications of the ACM, vol. 36, no. 7, pp. 74-84, July 1993.

[2] "Radio Frequency Identification: Applications and Implications for Consumers,"

Workshop Report from the Staff of the Federal Trade Commission, Federal Trade Commission, Mar. 2005.

[3] N. Borselius, "Mobile Agent Security," Electronics and Communication Engineering Journal, vol. 14, no. 5, pp. 211-218, Oct. 2002.

[4] A. Juels, "RFID Security and Privacy: A Research Survey," IEEE Journal of Selected Areas in Communications, vol. 24, no. 2, pp. 381-394, Feb. 2006.

[5] S.A. Weis, S.E. Sarma, R.L. Rivest, and D.W. Engels, "Security and Privacy Aspects of Low-Cost Radio Frequency Identification Systems," Security in Pervasive Computing 2003, LNCS 2802, pp. 201-212, 2004.

[6] K.W. Rhee, J. Kwak, S.J. Kim, and D.H. Won, "Challenge-Response Based RFID Authentication Protocol for Distributed Database Environment," International Conference on Security in Pervasive Computing 2005, LNCS 3450, pp. 70-84, 2005.

[7] S.M. Lee, Y.J. Hwang, D.H. Lee, and J.I. Lim, "Efficient Authentication for Low-Cost RFID Systems," International Conference on Computational Science and its Applications 2005, LNCS 3480, pp. 619-627, 2005.

[8] ISO/IEC 18000-6 Amd1, "Information technology - Radio frequency identification for item management - Part 6:Parameters for air interface communications at 860 MHz to 960 MHz, AMENDMENT 1: Extension with Type C and update of Types A and B," 2004.

[9] 정보통신단체표준(잠정표준) TTAI.KO-12.0091, "수동형 RFID 보안태그와 리더의 인증 및 데이터 보호 프로토콜," 2008년 12월.

[10] 최용제, 최두호, 이상연, 정교일, "수동형 RFID를 위한 보안 기술 구현," 한국통신학회 하계종합학술발표회, pp. 96-99, 2008년 7월.

〈著者紹介〉

양 연 형 (Yeonl Hyun Yang) 학생회원
2002년 2월: 포항공과대학교 전자전기공학과 졸업
2002년 3월~현재: 포항공과대학교 전자전기공학과 박사과정
〈관심분야〉 정보보호, 암호 이론

김 선 영 (Sun Young Kim) 학생회원
2008년 2월: 한동대학교 전산전자공학부 졸업
2008년 3월~현재: 포항공과대학교 정보통신대학원 석사과정
〈관심분야〉 암호학, 정보보호

이 필 중 (Pil Joong Lee) 종신회원
1974년 2월: 서울대학교 전자공학과 졸업
1977년 3월: 한국대학교 전자공학과 석사
1982년 6월: U.C.L.A System Science. Engineer
1985년 6월: U.C.L.A Electrical Engineering. Ph.D
1980년 3월~1985년 8월: Jet Propulsion Laboratory. Senior Engineer
1985년 8월~1990년 2월: Bell Communication Research. M.T.S
1990년 2월~현재: 포항공과대학교 전자전기공학과 교수
1996년 2월~1997년 2월: NEC Research Institute 방문 연구원
2000년 9월~2003년 8월: 포항공과대학교 정보통신 연구소장 (정보통신 대학원장 겸임)
2004년 1월~2004년 12월: 한국정보보호학회 회장
2004년 1월~2004년 12월: KT 정보보호 자문위원
2008년 7월~2008년 12월: POSDATA 정보보호 자문위원
2007년 1월~현재: 한국공학한림원 정회원
〈관심분야〉 정보보호 전반