情報保護學會論文誌
第19卷 第6號, 2009. 12

경량화된 MAC을 이용한 강력한 Yoking-Proof 프로토콜

조 창 현,† 이 재 식, 김 재 우, 전 문 석‡
숭실대학교

Strong Yoking-Proof Protocol using Light-Weighted MAC

Chang-hyun Cho,† Jae-sik Lee, Jae-woo Kim, Moon-seog Jun‡
Soongsil University

요 약

RFID 시스템을 이용하여 다수의 태그를 동시에 인증하기 위한 방법으로 Ari Juels는 Yoking-Proof 프로토콜을 제안하였다. 일반적인 Yoking-Proof 방법은 MAC(Message Authentication Code)을 이용하여 인증을 하기 때문에, 저가형 태그에 적용하기가 어렵다. MD5와 같은 일반적인 해쉬함수도 저가형 태그에 구현하는 것은 쉽지 않기 때문이다. 따라서 Ari Juels는 한번만 인증가능한 경량화된 Yoking-Proof 방법도 함께 제안하고 있다. 하지만 제안된 방법에서 사용된 경량화된 MAC인 Minimalist MAC은 그 특성상 한번만 사용가능하고, 또한 제안된 구조는 재생공격에 취약한 구조를 가지고 있다. 따라서 본 논문에서는 Lamport가 제안한 디지털 서명기법을 이용한 경량화된 MAC을 이용하여, 기존에 제안된 논문보다 재생공격에 강하고 다수의 키 값을 저장할 수 있는 형태의 태그를 이용한 Yoking-Proof 프로토콜을 제안한다.

ABSTRACT

Ari Juels proposed Yoking-Proof protocol for authenticating multiple tags simultaneously using RFID system. Because common Yoking-Proof methods authenticate by using MAC (Message Authentication Code), it is difficult to apply them to inexpensive tags. It is also difficult to implement common hash functions such as MD5 in inexpensive tags. So, Ari Juels also proposed a lightweighted Yoking-Proof method with only 1 authentication. However, Minimalist MAC, which is a lightweighted MAC used in the proposed method is for single-use, and the proposed structure is vulnerable to replay attacks. Therefore, in this study, the minimalist MAC using Lamport's digital signature scheme was adopted, and a new type of Yoking-Proof protocol was proposed where it uses tags that are safe from replay attacks while being able to save multiple key values.

Keywords: RFID, Authentication, Yoking-Proof, minimalist MAC

I. 서 론

RFID(Radio Frequency Identification)는 라디오 주파수를 이용하여 다수의 개체를 인식하는 기술로, ID를 부여 받은 개체인 태그와 태그를 읽는 리더, 그리고 ID정보를 관리하는 데이터베이스로 구성된다. RFID 태그는 Active 태그와 Passive 태그의 두 가지 종류로 나누어진다[1]. 제안하는 프로토콜은 Passive 태그의 인증에 관하여 다룬다. RFID 태그는 그 구조상 인증의 절차에 있어서 보안이 적용되지 않았을 경우 취약한 구조를 가지고 있다. 한정된 태그의 능력으로 인하여 태그와 리더 사이의 인증은 기술

접수일(2009년 7월 29일), 수정일(2009년 10월 19일),
게재확정일(2009년 11월 20일)
† 주저자, hist0001@hanmail.net
‡ 교신저자, mjun@ssu.ac.kr

적인 사항과 비용적인 사항을 고려하여 설계되어야 한다. 태그와 리더 사이의 인증기술 중 동시에 하나의 태그를 인증하는 것이 아닌, 다수의 태그를 한꺼번에 인증하는 여러 가지 기법들이 제안되었다[2-8]. 그 중 Ari Juels는 다수의 태그를 효율적으로 인증하기 위한 방법으로 Yoking-Proof를 이용한 인증하는 방법을 두 가지 제안하였다[2]. 첫 번째 방법인 일반적인 MAC(Message Authentication Code)를 이용한 Yoking-Proof 방법은 태그가 MD5와 같은 해쉬 함수를 이용한 MAC 함수를 이용하고 있다. 하지만 태그의 MAC 함수 구현은 현재 상용화된 태그들에 적용하기가 비용적인 면에서 많은 어려움이 있다. 현재 RFID 태그의 표준이라 할 수 있는 EPC Global Gen2 태그의 경우도 간단한 액세스 및 Kill 명령어와 CRC체크 정도만을 지원하고 있으며, 옵션으로 랜덤 함수를 제공할 수 있다고 명시하고 있을 뿐이다[9]. Ari Juels의 두 번째 Yoking-Proof 방법인 One-Time Yoking-Proof 프로토콜은 태그가 MAC을 이용하지 않고 인증을 할 수 있는 하나의 방법이 될 수 있다. 두 번째 방법의 경우 태그와 리더 사이의 Handshake 과정을 간소화 하고, MD5등과 같은 일반적인 해쉬 함수를 이용한 MAC함수가 아닌 Lamport Digital Signature를 이용한 경량화된 MAC인 minimalist MAC을 이용하여 MAC함수를 대체하여 사용하고 있다[10].

하지만 One-Time Yoking-Proof 프로토콜은 그 구조상 재생공격에 취약한 구조를 가지고 있으며[3], minimalist MAC을 이용하기 때문에 처음 인증에서 실패한 경우 두 번째 부터는 보안상 취약한 구조를 지니고 있다. 이러한 구조적 취약성을 개선하기 위하여 타임스탬프를 이용한 방법[3], 랜덤값을 이용한 방법[5-7], 랜덤값과 타임스탬프 모두를 이용한 방법[8]등 다양한 논문이 제안되었지만, 제안된 논문들은 일반적인 MAC함수를 이용하고 있다. 일반적인 MAC함수를 이용하는 경우, Ari Juels가 제안한 Yoking-Proof의 Standard 버전을 이용하여도 재생공격에 안전한 구조를 가지고 있다.

제안하는 프로토콜에서는 경량화된 MAC함수를 이용하여 재생공격에 강하면서, 한번만 인증할 수 있는 형태가 아닌 태그에 저장된 비밀키의 개수만큼 인증이 가능한 경량화된 강력한 Yoking-Proof 인증 프로토콜을 제안한다, 또한 두 개 이상의 다중 태그를 동시에 인증할 수 있는 구조로 확장이 가능하다. 본 논문의 구성은 다음과 같다. 우선 2장에서 기존에 제안된 Yoking-Proof와 관련된 연구들을 간단히 기술하고, 3장에서는 제안하는 경량화된 MAC을 이용한 강력한 Yoking-Proof 프로토콜에 대하여 설명한다. 4장에서는 제안하는 프로토콜의 보안성의 분석 및 발생할 수 있는 문제점을 살펴보고 이를 해결하기 위한 방법을 제시하고, 5장에서 결론을 맺는다.

II. 기존의 Yoking-Proof 인증 관한 관련 연구

2.1 용어 및 표기 설명

앞으로 설명할 프로토콜을 위하여 사용될 용어 및 표기는 다음과 같다.

- T_i : RFID 태그 i를 나타낸다.
- $A, B, \ldots, N$: RFID 태그의 ID값을 나타낸다. (예를 들어, A인 경우 T_A의 ID값임)
- C_i : T_i의 카운터 값이다
- X_i : T_i의 비밀키 값이다.
- IV_i : T_i가 가지고 있는 최초의 랜덤값이다.
- TS : 타임스탬프값으로 데이터베이스나 검증자가 생성하는 값이다.
- $\triangle t$: 다수의 태그를 인증하여 검증하는 과정은 사전에 정의된 $\triangle t$ 시간 안에 이루어져야 한다.
- f : 비밀키를 이용한 해쉬 함수로 일종의 MAC이라 할 수 있다.
- $MAC_{k_i}[v_1, v_2, \cdots, v_n]$: k_i를 이용하여 v_1, v_2, …, v_n 의 MAC 값을 생성하는 함수로 MD5와 같은 일반적인 해쉬함수를 이용하는 MAC 이다.
- $mMAC_{k_i}[v_1, v_2, \cdots, v_n]$: Lamport의 Digital Signature[10]을 이용하여 사전에 정의된 키 값 k_i을 써서 이용하여 MAC 값을 생성하는 함수로 경량화된 MAC이다.(이후 $mMAC$으로 표기) $mMAC$은 0과 1의 key 비트를 이용하여 사전에 정의된 인증코드 값이 생성되는 구조를 가지고 있다. 예를 들어 8비트의 출력을 표현하는 $mMAC$이 있다고 하자. 이때 키 값 X_A가 [표 1]과 같고, 함수의 입력 값으로 “11001001”이 들어오면 출력은 “11111010”이 된다. 즉, $mMAC_{X_A}(11001001_2) = 11111010_2$ 이다.

[표 1] X_A의 MAC Table

X_A	1	2	3	4	5	6	7	8
bit 0	1	0	1	1	0	0	1	1
bit 1	1	1	0	1	1	1	1	0

- P_{AB} : 최종적으로 리더가 검증자에게 검증을 할 때 전송하는 데이터로 인증을 위한 정보들이 포함된다.
- V : 검증자로 태그의 정보를 가지고 있는 데이터베이스에 접근이 가능하다.
- α / β : Left 및 Right Proof 과정을 통하여 얻어진 결과 값들의 모음을 나타내기 위한 용어이다.

2.2 프로토콜의 일반적인 과정 및 가정

Yoking-Proof 프로토콜은 다음과 같은 4가지 과정을 통하여 다수의 태그를 검증 할 수 있다.

과정 1: Left Proof 과정으로 첫 번째 태그 인증이 일어나는 부분을 Left Proof 라고 한다.

과정 2: Right Proof 과정으로 두 번째 혹은 그 이상의 태그 인증이 일어나는 부분을 Right Proof 라 한다. 즉 두 개 이상의 다수의 태그를 인증하는 경우 두 번째 이후의 모든 태그를 인증하는 부분은 이전의 Right Proof 과정과 동일하다.

과정 3: Last Proof 과정으로 다수의 태그를 인증하기 위하여, Left Proof를 요청했던 태그에 마지막으로 다시 한번 인증을 요청하는 부분을 Last Proof라 한다. 프로토콜에 따라서 Last Proof과정이 없이, Left Proof와 Right Proof로만 이루어지는 인증도 있다.

과정 4: 과정 3으로부터 얻어진 인증정보를 이용하여 리더는 검증자에게 태그들을 검증한다.

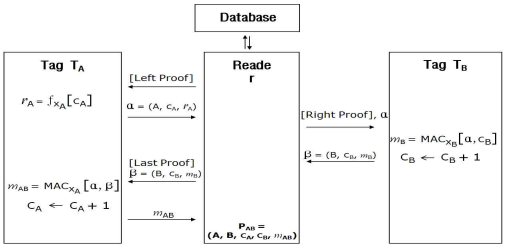
앞으로 살펴볼 태그와 리더 사이의 과정은 다음과 같은 사항을 가정하고 있다.

- 리더와 데이터베이스 사이는 안전한 채널이라고 가정하며, 데이터베이스를 검증자로 볼 수 있다.
- 리더와 태그 사이는 불안전한 채널이라고 가정한다.
- 모든 리더는 검증자에게 검증요청을 할 수 있다.
- 태그의 비밀키 값인 X_i는 데이터베이스와 사전에 공유하고 있다.
- 태그는 데이터베이스와 최초의 랜덤값인 IV_i를 공유하고 있다.
- 검증과정은 $\triangle t$ 안에 이루어져야 하며, 이 값은 적용되는 시스템에 따라서 다르게 정의된다.

2.3 관련 연구

2.3.1 표준 암호를 이용한 Yoking-proof 프로토콜

Ari Jules가 제안한 Yoking-Proof의 일반적인



[그림 1] Yoking-proof 프로토콜(표준 암호 버전)

MAC을 이용하는 프로토콜은 [그림 1]과 같다[2]. 이 프로토콜은 T_A와 T_B를 동시에 인증하는 구조로 카운터값 및 f함수를 이용한 랜덤값 생성과 MAC함수를 이용하는 특징이 있다. 프로토콜의 전체적인 인증과정은 다음과 같다.

과정 1: 리더는 T_A에 Left Proof를 요청한다. T_A는 C_A를 이용하여 r_A를 생성한다. r_A는 T_A의 비밀키인 X_A를 이용한 해쉬(f)로 생성되기 때문에, X_A를 알고 있는 T_A만 생성할 수 있다. T_A는 생성된 값을 포함한 α를 리더에게 전달한다. (여기서 α는 A, C_A, r_A이다.)

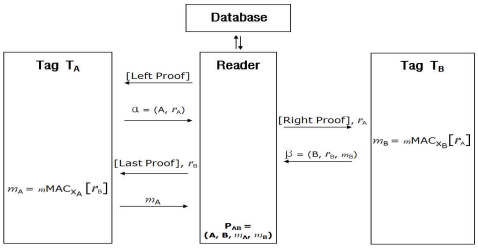
과정 2: 리더는 T_B에 Right Proof를 요청하고, T_A에서 전달받은 α를 전달한다.T_B는 α와 C_B를 비밀키 X_B를 이용하여 MAC 값인 m_B를 생성을 한다. T_B는 생성된 값들을 포함한 β를 리더에게 전달한다. (여기서 β는 B, C_B, m_B이다.) 그리고 T_B는 자신의 카운터 값인 C_B를 1 증가 시킨다.

과정 3: 리더는 T_A에게 Last Proof를 요청하고, β를 다시 T_A에게 전달한다. T_A는 비밀키 X_A를 이용하여 α와 β의 MAC 값인 m_{AB}를 생성하고, 이를 리더에 전달한다. 그리고 T_A는 자신의 카운터 값인 C_A를 1 증가시킨다.

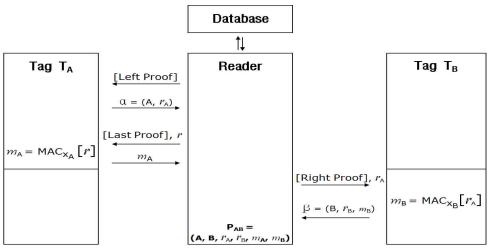
과정 4: 리더는 T_A와 T_B가 동시에 인식되어 인증되었다는 것을 검증하기 위하여 검증자에게 P_{AB}를 전달한다. (여기서 P_{AB}는 A, B, C_A, C_B, m_{AB}이다.) 검증자는 $\alpha' = (A, C_A, f_{X_A}[C_A])$ 와 $\beta' = (B, C_B, MAC_{X_B}[\alpha', C_B])$를 계산하여, $P_{AB}' = MAC_{X_A}[\alpha', \beta']$를 계산할 수 있다. 만약 검증자가 생성한 MAC 값인 P_{AB}'가 리더가 전달한 P_{AB}와 같다면, T_A와 T_B는 동시에 인증된 것이다.

2.3.2 경량화된 MAC을 이용한 One-Time Yoking-Proof 프로토콜

Yoking-Proof의 경량화된 MAC을 이용하는 프



[그림 2] One-time Yoking-proof 프로토콜 (경량 암호 버전)



[그림 3] One-time Yoking-Proof 프로토콜 구조에서의 재생공격

로토콜은 [그림 2]와 같다[2]. 2.2.1과 같이 T_A와 T_B를 동시에 인증하는 구조로 $mMAC$이라는 경량화된 MAC을 이용하여, 한번만 인증을 할 수 있는 특징을 가지고 있다[2, 10]. 프로토콜의 전체적인 인증 과정은 다음과 같다.

과정 1: 리더는 T_A에 Left Proof를 요청한다. T_A는 미리 저장된 r_A를 포함한 α를 리더에게 전달한다. 여기서 r_A는 T_A가 생성하는 것이 아니고, 사전에 생성된 랜덤값이 태그에 저장된 것으로 데이터베이스에도 동일한 값이 저장되어 있다. (여기서 α는 A, r_A이다.)

과정 2: 리더는 T_B에 Right Proof를 요청하고, r_A를 T_B에 전달한다. T_B는 $mMAC$와 비밀키 X_B를 이용하여 r_A의 MAC 값인 m_B를 생성을 한다. 그리고 사전에 생성된 랜덤값인 r_B를 포함한 β를 리더에 전송한다. (여기서 β는 B, r_B, m_B이다.)

과정 3: 리더는 T_A에게 Last Proof를 진행한다고 알려주며, r_B를 T_A에 전달한다. T_A는 T_A의 비밀키 X_A를 이용하여 경량화된 MAC함수인 $mMAC$을 사용해서 인증코드 m_A를 생성하고 이를 리더에 전달한다.

과정 4: 리더는 T_A와 T_B가 동시에 인식되어 인증되었다는 것을 검증하기 위하여 검증자에게 P_{AB}를 전달한다. (여기서 P_{AB}는 A, B, m_A, m_B이다.) 검증자는 $m_B' = mMAC_{X_B}[r_A]$와 $m_A' = mMAC_{X_A}[r_B]$을 계산하여 P_{AB}와 비교하여 최종 인증을 할 수 있다.

2.3.3 타임스탬프를 이용한 Yoking-Proof 프로토콜

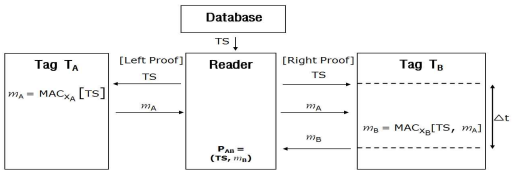
SAITO와 SAKURAI는 Ari Juels가 제안한 Yoking-Proof의 2번째 방법인 One-time Yoking-Proof가 구조적으로 재생공격에 취약하다는 것을 발견하였다[3]. 재생공격은 다음과 같은 상황을 이용하여 이루어진다. 1) 공격자는 악의적인 리더이며 검증자에 검증요청이 가능한 권한을 가지고 있다. 2) 악의적인 리더는 두 개의 정상적인 태그를 동시에 인증하는 것이 아니고, 하나의 정상적인 태그를 이용하여 비정상적인 태그를 함께 인증을 한다. 3) 이를 위하여 악의적인 리더는 사전에 정상적인 태그의 인증정보를 수집하고 이를 이용한다.

실제로 One-time Yoking-Proof 프로토콜의 구조에서 재생공격이 어떻게 일어나는지 살펴보면 [그림 3]과 같다.

악의적인 목적을 가진 리더는 T_A에 Left Proof 과정을 통하여 정상적인 α인 A와 r_A를 전달 받는다. 이후 리더는 Right Proof 과정을 진행해야 하지만, 악의적인 리더는 Last Proof 과정을 먼저 진행한다. 단, 이때 사용하는 랜덤값은 악의적인 리더가 임의로 생성한 랜덤값 r이다. T_A는 악의적인 리더로부터 전달받은 r이 T_B가 생성한 값으로 착각하고 Last Proof 과정을 진행하여 m_A를 생성하여 악의적인 리더에 전달한다. 악의적인 리더는 이후, 두 개의 정상적인 태그가 아닌 하나의 정상적인 태그만 있어도 인증이 가능하다. 즉, 정상적인 태그에게 기존에 수집했던 정보인 r_A를 전송하여 Right Proof 과정을 진행한다. [그림 3]과 같이 T_B에게 Right Proof를 요청하였을 경우, T_B는 β인 B, r_B, m_B를 악의적인 리더에 전송한다. 악의적인 리더는 사전에 입수한 정보인 A와 m_A를 이용하여, P_{AB}를 검증자에게 전송하여 A와 B 모두를 함께 인증 받는다. (여기서 P_{AB}는 A, B, m_A, m_B이다.)

[그림 3]과 같은 재생공격이 가능한 이유는 1) Last Proof 과정에서 T_A는 r_B를 이용하여 m_A를 생성해야 하지만, T_A는 입력받은 랜덤값이 실제 T_B가 생성한 값인지 확인할 수 없기 때문이다. 2) 또한, 매 인증과정마다 동일한 T_A의 인증값을 사용해도 되기 때문이다.

이러한 취약점 때문에 재생공격을 피하기 위하여 타임스탬프값을 이용한 Yoking-Proof가 제안되었다



[그림 4] 타임스탬프를 이용한 Yoking-Proof 프로토콜

[3]. [그림 4]는 타임스탬프를 이용한 프로토콜의 과정을 보여주며, 이 프로토콜은 Last Proof 과정이 없는 구조로 일반적인 MAC함수를 이용한다.

과정 1: 리더는 데이터베이스에 TS를 전송받는다. 그리고 T_A에 Left Proof 를 요청하며 TS를 전달한다. T_A는 비밀키 X_A를 이용하여 TS의 MAC 값인 m_A를 생성하고 리더에 전달한다.

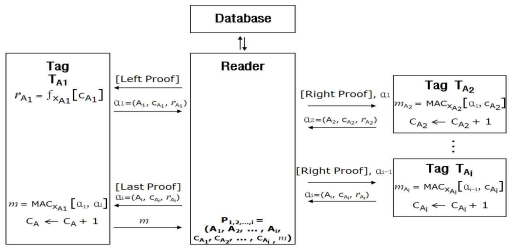
과정 2: 리더는 T_B에 Right Proof 를 요청하며 TS와 T_A로부터 전달받은 m_A를 전달한다. T_B는 비밀키 X_B를 이용하여 TS와 m_A의 MAC 값인 m_B를 생성하여 리더에 전송한다.

과정 3: 리더는 P_{AB}를 검증자에게 전송하여 T_A와 T_B를 인증한다. 검증자는 $m_A' = MAC_{X_A}[TS]$와 $m_B' = MAC_{X_B}[TS, m_A']$를 계산하여 m_B와 비교하여 리더가 전달한 P_{AB}와 같고, TS와 $\triangle t$를 이용하여 $(TS+\triangle t)$ 시간 안에 검증이 이루어지면 두 개의 태그는 동시에 인증이 된 것이다.

저자는 또한 두개 이상의 다수의 태그를 동시에 인증하기 위한 방법도 제시하고 있으며, 이는 2.3.4절에서 설명하는 방법과 비슷하며, Right Proof 과정이 계속해서 진행되는 형태이므로 자세한 내용은 참고문헌 [3]을 참조하기 바란다.

2.3.4 그룹 인증을 위한 Yoking-Proof 프로토콜

Leonid Bolotnyy와 Gabriel Robins는 Ari Juels가 제안한 Yoking-Proof의 그룹버전을 제안하였다[4]. Ari Juels의 Yoking-Proof[2]는 두 개의 태그를 동시에 인증하는 프로토콜이다. 따라서 두 개 이상의 다수의 태그를 동시에 인증하기 위해서는 [그림 5]와 같이 Right Proof 과정이 계속해서 진행되어야 한다. 그 외에 나머지 과정은 2.3.1과 동일하므로 프로토콜의 자세한 설명은 생략한다.



[그림 5] 그룹 인증 Yoking-Proof 프로토콜

2.3.5 랜덤값을 이용한 그룹 인증 프로토콜

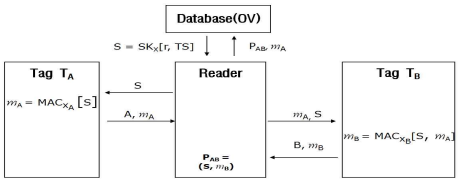
Selwyn Piramuthu는 타임스탬프를 이용한 방법 또한 재생공격으로부터 안전하지 않다는 것을 보였다[5]. 이는 타임스탬프값은 순차적으로 증가하는 성질로 인하여 그 값의 유추가 가능하기 때문에 사전에 타임스탬프값을 유추하여 재생공격을 하는 경우 여전히 재생공격으로부터 안전하지 않기 때문이다. 따라서 타임스탬프와 같이 유추 가능한 값이 아닌 랜덤한 값을 타임스탬프값 같이 이용하는 방법을 제안하였다[5]. 랜덤값을 이용한다는 것을 제외한 나머지구조는 타임스탬프를 이용한 구조와 동일하므로 자세한 내용은 참조논문 [5]를 살펴보기 바란다.

2.3.6 두 개의 랜덤값을 이용한 인증 프로토콜

하나의 랜덤값이 아닌 2개의 랜덤값을 이용하여 재생공격을 수행하였을 경우 그 복잡성을 높인 논문이 제안되었다[6,7]. 하지만 이 방법은 기존에 제안된 하나의 랜덤값을 이용하는 방법[5]과 비교하여 복잡성이 증가되었지만, 인증과정에 사용되는 랜덤값의 크기가 큰 경우 저장 공간 및 통신횟수의 비교는 무의미하다. 예를 들어 랜덤값이 32비트인 경우 저장 공간은 2^{32}비트 이상이 되며, 통신횟수 또한 2^{32}회 이상의 통신을 해야 한다. 이는 약 4억번이 넘는 통신횟수를 필요로 하며, 이는 하나의 태그와 통신하는데 5ms의 시간이 소요된다고 가정했을때 약 6천시간 정도의 통신이 필요하다. 즉, 무차별적인 공격을 통한 재생공격은 현실적으로 불가능하다.

2.3.7 SecTS-Proof 프로토콜

타임스탬프와 랜덤값을 모두 이용하는 Yoking-Proof 방법도 제안되었다[8]. [그림 6]은 랜덤값과 타임스탬프값을 검증자의 비밀키를 이용하여 메시지 인증코드인 S를 생성하고 이를 타임스탬프를 이용한 구조처럼 사용하는 과정을 보여준다. 타임스탬프나 랜



[그림 6] SecTS-Proof 프로토콜

덤값 대신 검증자로부터 생성된 S를 이용한다는 점이 기존의 논문들[3,5-7]과 다른점이며 나머지 구조는 기존의 논문들과 동일하다. 참고문헌 [8]의 저자는 기존의 랜덤값을 이용하는 방법인 참고문헌 [5]가 레이스 컨디션 공격에 취약하다고 주장하며, 자신이 제안한 논문은 재생공격을 포함한 레이스 컨디션 공격을 막을 수 있다고 하였으나, 검증과정에서 이를 제시하지 않고 있다.

레이스 컨디션공격은 태그나 리더가 동시에 여러 개의 세션을 받아들이면서 나타나는 문제이기 때문에, 태그의 경우 한 번에 하나의 리더에만 응답하면 피할 수 있으며, 리더의 경우도 마찬가지이다. 한 번의 응답은 400ms 이내에 이루어져야 하기 때문에[2,11], 동시에 응답하지 않아서 발생할 수 있는 시간적인 딜레이는 매우 낮다고 할 수 있다.

III. 제안하는 프로토콜

앞에서 살펴본 논문들은 Ari Juels의 One-time Yoking-Proof 방법[2]을 제외한 나머지 방법[3-8]이 일반적인 MAC함수를 이용하고 있다. 하지만 MAC함수의 구현은 저가형 태그에서는 많은 게이트 수가 필요하기 때문에 구현이 쉽지 않다. EPC Global의 Class1 UHF Gen2의 표준을 살펴보면 태그에 접근하기 위한 패스워드 및 Kill명령을 위한 패스워드와 간단한 CRC만 제공하고 있다. 또한 옵션을 통하여 랜덤값을 생성하는 부분을 남겨 놓고 있을 뿐, MD5와 같은 해쉬함수의 구현은 제공하지 않는다[9]. 따라서 본 논문에서는 Lamport의 Digital Signiture 기반의 MAC[10]을 이용하여 기존에 알려진 재생공격에 강한 프로토콜을 제안한다. 하지만 참고문헌[10]에서 제안된 방법은 일회용 MAC함수로 한번 이상 사용할 경우 그 구조상 비밀키 값이 노출되는 취약점을 가지고 있다. 따라서 본 논문에서는 태그에 미리 복수 개의 비밀키를 저장하여, 매 인증마다 다른 비밀키를 이용하여 메시지인증코드인 MAC값을 생성한다.

3.1 제안하는 프로토콜의 환경

제안하는 프로토콜은 사전에 태그와 데이터베이스 사이에 복수의 비밀키 값이 정의되어 있어야 한다. 또한 비밀키 값의 길이와 개수는 태그의 용량 및 적용환경에 따라 달라져야 한다. Ari Juels는 One-time Yoking-Proof 프로토콜에서 랜덤값은 120비트, 비밀키의 길이는 240비트를 사용할 것을 권장하고 있다. 따라서 총 360비트의 저장 공간이 mMAC을 사용하는 Yoking-Proof에서 추가로 필요하다[2]. mMAC은 구조상 입력과 출력의 2배에 해당하는 비밀키 값이 필요하기 때문이다. [표 2]는 태그의 저장용량을 나타내며, [표3]은 데이터베이스의 저장공간을 보여준다.

초기 랜덤값인 r_A의 크기가 n이고, 비밀키 값인 X_{A_1}의 값이 2n 인 경우, 총 k개의 비밀키를 가지고 있는 태그는 경우 총 (n+2n*k)비트의 저장 공간이 필요하

[표 2] T_A의 용량 계산

항목	r_A	X_{A_1}		X_{A_2}		…	X_{A_k}		합계 (bit)
		bit 0	bit 1	bit 0	bit 1	…	bit 0	bit 1	
필요비트수	n	n	n	n	n	…	n	n	n+2n*k

[표 3] 데이터베이스 구조

T	r	C	X_1	X_2	…	X_k	…
T_A	r_A	C_A	X_{A_1}	X_{A_2}	…	X_{A_k}	…
T_B	r_B	C_B	X_{B_1}	X_{B_2}	…	X_{B_k}	…
⋮	⋮	⋮	⋮	⋮	…	⋮	…
T_i	r_i	C_i	X_{i_1}	X_{i_2}	…	X_{i_k}	…

다.

데이터베이스에도 각 태그의 초기 랜덤값 r과, 비밀키 값의 현재 위치를 나타내는 카운터 값 C, 그리고 다수의 비밀키인 X_1, X_2, … , X_k가 저장되어 있다. 여기서 카운터 값 C는 사전에 0으로 모두 초기화 되며, 카운터 값이 비밀키의 마지막 위치인 k과 같아지면, 그 다음 카운터 값은 다시 0으로 초기화 된다. 랜덤값 r과 비밀키 X_1, X_2, … , X_k는 태그의 공급자가 사전에 미리 값을 생성하여 태그와 데이터베이스에 저장한다.

3.2 제안하는 프로토콜의 구조

제안하는 프로토콜의 기본 구조는 Ari Juels의 Standard Yoking-Proof와 같은 구조를 가지고 있으며, 경량화된 MAC인 mMAC을 이용하는 것이 특징이다. [그림 7]은 제안하는 프로토콜의 구조를 보여준다.

제안하는 프로토콜의 각 과정은 다음과 같이 10가지 세부적인 단계로 나누어진다. 또한 두 개 이상의 태그를 동시에 인증하기 위해서는 Yoking-Proof 프로토콜의 2번째 과정인 Right Proof 과정에 해당하는 단계 3~5의 과정을 반복적으로 수행하면 된다.

단계 1: 리더는 T_A에 Left Proof를 요청한다.

단계 2: T_A는 미리 저장된 r_A를 포함한 α를 리더에 전송한다. (여기서 α는 A, C_A, r_A 이다.)

단계 3: 리더는 T_B에 Right Proof를 요청하고, T_A에서 전달받은 인자 중 r_A를 함께 전달한다.

단계 4: T_B는 비밀키 X_B를 이용하여 r_A의 MAC 값인 m_B를 생성한다. 이때 사용된 MAC함수는 $mMAC$이다. T_B는 미리 저장된 r_B를 포함한 β를 리더에 전달한다. (여기서 β는 B, C_B, r_B, m_B이다.)

단계 5: T_B는 자신이 가지고 있는 랜덤값 r_B을 다음과 같이 업데이트 하고, 현재 사용된 비밀키 X_B를 다음 비밀키 값으로 업데이트 한다. 그리고 마지막으로 비밀키의 위치를 나타내는 카운터 C_B를 1 증가한다. 각 업데이트 과정은 다음과 같다.

$r_B = mMAC_{X_B}[m_B]$,

$X_{B_{C_B+1}} \leftarrow X_{B_{C_B}}$, $C_B \leftarrow C_B + 1$

단계 6: 리더는 T_A에 Last Proof를 요청하고, 단계 4에서 T_B로부터 전달받은 인자 중 r_B를 함께 전달한다.

단계 7: T_A는 비밀키 X_A를 이용하여 r_B의 MAC 값인 m_A를 생성하고 이를 리더로 전달한다. 이때 사용된 MAC함수는 $mMAC$이다.

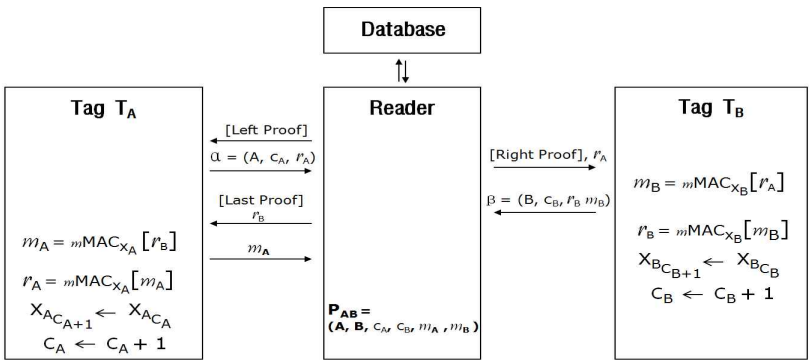
단계 8: T_A는 자신이 가지고 있는 랜덤값 r_A을 다음과 같이 업데이트 하고, 현재 사용된 비밀키 X_A를 다음 비밀키 값으로 업데이트 한다. 그리고 마지막으로 비밀키의 위치를 나타내는 카운터 C_A를 1 증가한다. 각 업데이트 과정은 다음과 같다.

$r_A = mMAC_{X_A}[m_A]$, $X_{A_{C_A+1}} \leftarrow X_{A_{C_A}}$, $C_A \leftarrow C_A + 1$

단계 9: 리더는 단계 7에서 T_A로부터 전달받은 m_A를 포함한 P_{AB}를 검증자에게 전달한다. (여기서 P_{AB}는 A, B, C_A, C_B, m_A, m_B이다.)

검증자는 $m_A' = mMAC_{X_A}[r_B]$와 $m_B' = mMAC_{X_B}[r_A]$를 계산하여 P_{AB}의 m_A, m_B와 비교하여 동일한 경우 T_A와 T_B가 동시에 인식되어 인증되었다는 것을 검증할 수 있다.

단계 10: 데이터베이스는 단계 9에서 검증이 올바르게 완료 된 경우, T_A와 T_B의 정보를 다음과 같이 업



[그림 7] 제안하는 프로토콜

데이트 한다. 이는 단계 5와 단계 8에서 T_A와 T_B가 업데이트 된 것과 동일한 과정이다.

$r_A = mMAC_{X_A}[m_A],\ C_A \leftarrow C_A + 1$

$r_B = mMAC_{X_B}[m_B],\ C_B \leftarrow C_B + 1$

IV. 보안 및 문제점 분석

4.1 재생공격에 대한 안정성

제안하는 프로토콜에 대해서 재생공격이 일어날 경우 다음과 같이 재생공격은 실패 한다.

[태그정보 수집단계]

공격 1: 공격자는 Left Proof 단계를 통하여 T_A의 정보인 $\alpha = (A, C_A, r_A)$를 얻는다.

공격 2: 공격자는 Last Proof 단계를 먼저 진행한다. 이를 위해서 자신이 임의로 생성한 랜덤값인 r_V를 T_A에게 전송하여, $m_A = MAC_{X_A}[r_V]$를 얻는다. 최종적으로 공격자는 인증과정에서 필요한 P_{AB} 정보 중 (A, C_A, m_A)를 수집할 수 있다.

[재생공격 실행단계]

공격 3: 공격자는 Right Proof 단계를 통하여 T_B에게 공격 1에서 수집한 r_A를 전송하여 $m_B = MAC_{X_B}[r_A]$를 포함한, $\beta = (B, C_B, r_B, m_B)$를 얻는다.

공격 4: 공격자는 $P_{AB} = (A, B, C_A, C_B, m_A, m_B)$을 모두 수집하였으며, 이를 이용하여 검증자에 검증을 시도한다.

[검증단계]

검증 1: 검증자는 데이터베이스에 저장되어 있는 T_A와 T_B의 정보를 이용하여 $m_A' = mMAC_{X_A}[r_B]$와 $m_B' = mMAC_{X_B}[r_A]$를 계산하고, 공격 4에서 전송받은 P_{AB}의 m_A, m_B와 비교한다.

검증 2: 검증 1은 데이터베이스의 r_B를 이용하여 m_A'를 생성한다. 하지만 공격 4로부터 전송받은 m_A는 공격자가 임의로 생성한 r_V를 이용하여 생성된 값이다. 따라서 $mMAC_{X_A}[r_B] \neq mMAC_{X_A}[r_V]$ 이므로, $m_A' \neq m_A$이다. 즉, 공격 3에서 T_B가 생성한 랜덤값 r_B를 이용하지 않기 때문에 재생공격은 실패하고, 전체적으로 T_A와 T_B는 동시에 인증되었다는 검증에 실패하게 된다.

4.2 태그와 데이터베이스 정보 동기화 문제

제안하는 프로토콜은 태그와 데이터베이스 사이에 카운터 정보 및 랜덤값 정보를 공유하고 있으며, 매 인증마다 업데이트 되는 특징을 가지고 있다. 따라서 태그의 정보만 업데이트 되었을 경우 또는 데이터베이스의 정보만 업데이트 되었을 경우 태그와 데이터베이스는 서로 다른 값을 저장하게 된다. 이러한 동기화 문제를 해결하기 위해서 다음과 같이 연속된 두 번의 인증과정을 통하여 해결할 수 있다.

가정: T_A와 T_B가 저장하고 있는 C_A, C_B, r_A, r_B가 데이터베이스가 저장하고 있는 값과 다르다.

과정 1: 리더는 $P_{AB} = (A, B, C_A, C_B, m_A, m_B)$를 검증자에게 전송하여 T_A와 T_B를 검증하길 원한다.

과정 2: 검증자는 우선 카운터 값인 C_A, C_B 를 데이터베이스가 저장하고 있는 값과 비교한다. 만약 카운터 값이 다른 경우, 검증자는 검증에 실패 했다는 메시지를 리더에 전달하고 검증정보는 잠시 보관하고 있는다. 카운터 값이 같다면, 검증자는 데이터베이스의 정보를 이용하여 m_A'과 m_B'를 생성하고, 리더로부터 전달받은 m_A, m_B와 비교한다. 만약 $m_A' \neq m_A$ 또는 $m_B' \neq m_B$인 경우 검증자는 검증에 실패 했다는 메시지를 리더에 전달하고 검증정보는 잠시 보관하고 있는다.

과정 3: 검증에 실패했다는 메시지를 전송받은 리더는 기존의 검증정보 이외에 새로운 검증정보를 다시 한번 수집한다. 새로운 검증정보는 $P_{AB^n} = (A, B, C_A + 1, C_B + 1, m_{A^n}, m_{B^n})$와 같이 표현한다.

과정 4: 리더는 검증자에게 연속된 검증 정보인 P_{AB}와 P_{AB^n}를 전송하고, 동기화가 잘못되었다고 알려준다. 검증자는 과정 2에서 잠시 저장해둔 P_{AB}와 과정 4로 전송받은 P_{AB}를 비교하여 동일한 경우, 동기화가 잘못되었다고 인식하고, 다음의 값을 계산한다.

$$r_A = mMAC_{X_A}[m_A],\ r_B = mMAC_{X_B}[m_B]$$

여기서 X_A와 X_B는 C_A, C_B 위치에 있는 비밀키 값이다. C_A, C_B, m_A, m_B는 리더로부터 전송받은 P_{AB}의 값이다. 그리고 다음 값을 계산한다.

$$m_{A^n}' = mMAC_{X_A}[r_B],\ m_{B^n}' = mMAC_{X_B}[r_A]$$

여기서 X_A와 X_B는 $C_A + 1$, $C_B + 1$ 위치에 있는 비밀키 값이다. 검증자는 계산된 값을 이용하여 $m_{A^n} = m_{A^n}'$이고, $m_{B^n} = m_{B^n}'$인 경우, 동기화가 잘못되

었으며 두 번의 인증과정을 통하여 올바르게 태그 T_A와 T_B가 인증되었다는 것을 검증한다. 그리고 데이터베이스의 정보 중 카운터 값은 C_A+1, C_B+1 로 저장하며, 랜덤값은 과정 4의 처음 계산에서 생성했던 r_A와 r_B로 저장한다.

위와 같은 두 번의 연속된 인증과정으로 태그의 검증 및 데이터베이스의 업데이트가 가능한 이유는 태그와 데이터베이스는 둘만이 알고 있는 공유된 비밀키 값을 가지고 있기 때문이다. 즉 정상적인 태그만 연속적으로 P_{AB}와 $P_{AB''}$를 생성할 수 있고, 이 값들은 과정 4를 통하여 데이터베이스에서 검증 가능하기 때문이다.

V. 결 론

다수의 태그를 동시에 인증하기 위한 기술로 Ari Juels의 One-time Yoking-Proof은 재생공격에 취약한 단점이 있다[3]. 이를 해결하기 위해서 타임스탬프를 이용한 방법[3], 랜덤값을 이용한 방법[5-7], 타임스탬프와 랜덤값을 이용한 방법[8]등이 제안되었으나 이 방법들은 모두 메시지인증코드 생성을 위하여 일반적인 MAC함수를 사용하고 있다. 하지만 태그의 MAC 함수 구현은 현재 상용화된 저가형 태그들에 적용하기 어렵다[9]. 따라서 제안하는 프로토콜은 경량화된 MAC함수인 mMAC을 이용하여 재생공격에 강력하고, 처음 인증에 실패한 경우 두 번째 부터는 보안상 취약한 구조가지고 있는 mMAC을 개선한 방법을 제안하였다. 이를 위해서 태그와 데이터베이스는 사전에 정의된 k개의 비밀키 값을 가지고 있으며, 카운터 값을 이용하여 현재 사용되고 있는 비밀키 값을 알 수 있도록 하였다. 또한 태그와 데이터베이스 사이에서 발생할 수 있는 동기화 문제를 해결하기 위한 방법을 제시하고 있다. 제안하는 프로토콜은 MAC함수를 구현할 수 없는 저가형 태그를 이용하여 재생공격에 강력한 인증을 할 수 있는 하나의 방법이 될 수 있다.

참 고 문 헌

[1] K. Finkenzeller, RFID Handbook : Fundamentals and Applications in Contactless Smart Cards and Identification, 2nd Ed., Wiley, May 2003.

[2] A. Juels, "Yoking Proofs for RFID Tags," IEEE Computer Society, Proceedings of the First International Workshop on Pervasive Computing and Communication Security-PerSec 2004, pp. 138-143, Mar. 2004.

[3] J. Saito and K. Sakurai, "Grouping Proof for RFID Tags," Proceedings of the 19th International Conference on Advanced Information Networking and Applications (AINA'05), pp. 621-624, Mar. 2005.

[4] L. Bolotnyy and G. Robins, "Generalized 'Yoking-Proof' for a Group of Radio Frequency Identification Tags," Technical Report CS-2006-12, University of Virginia, July 2006.

[5] S. Piramuthu, "On existence proofs for multiple RFID Tags," IEEE Computer Society Press. In IEEE International Conference on Pervasive Services, Workshop on Security, Privacy and Trust in Pervasive and Ubiquitous Computing - SecPerU 2006, Lyon, France, IEEE, pp. 317-320, June 2006.

[6] 조정식, 여상수, 김성권, "RFID Tag를 위한 강력한 Yoking Proof Protocols," 한국통신학회논문지, 32(3), pp. 310-318, 2007년 3월.

[7] J. Cho, S. Yeo, S. Hwang, S. Rhee, and S. Kim, "Enhanced Yoking proof protocols for RFID tags and tag groups," 22nd International Conference on Advanced Information Networking and Applications, pp. 1591-1596, Mar. 2008.

[8] C.C. Lin, Y.C. Lai, J.D. Tygar, C.K. Yang, and C.L. Chiang, "Coexistence Proof Using Chain of Timestamps for Multiple RFID Tags," In Proceedings of Advances in Web and Network Technologies and Information Management, LNCS 4537, pp. 634-643, 2007.

[9] EPC Global Inc., "Class 1 Generation 2 UHF Air Interface Protocol Standard Gen 2," Version 1.2.0, Oct. 2008.

[10] L. Lamport, "Constructing digital signatures from a one way function," Technical Report CSL-98, SRI International, Oct. 1979.

[11] AutoID Labs, "860 MHz-960 Mhz class 1 radio frequency identification tag radio frequency and logical communication interface specification recommended standard," Technical Report MIT-AUTOID -WH-007, Auto-ID Labs, Dec. 2003.

<著者紹介>

조 창 현 (Chang-hyun Cho) 학생회원
2000년 2월: 남서울대학교 정보통신공학과 졸업
2004년 2월: 숭실대학교 산업기술정보대학원 전자 및 컴퓨터공학과 석사
2004년 3월 ~ 현재: 숭실대학교 컴퓨터학과 박사과정
<관심분야> RFID, 정보보호, 네트워크 보안

이 재 식 (Jae-sik Lee) 학생회원
2005년 8월: 경원대학교 컴퓨터공학과 졸업
2007년 8월: 숭실대학교 컴퓨터학과 석사
2007년 9월 ~ 현재: 숭실대학교 컴퓨터학과 박사과정
<관심분야> RFID/USN, 인증이론 및 시스템, 암호프로토콜

김 재 우 (Jae-woo Kim) 학생회원
2007년 2월: 서울산업대학교 컴퓨터공학과 졸업
2009년 2월: 숭실대학교 컴퓨터학과 석사
2009년 3월 ~ 현재: 숭실대학교 컴퓨터학과 박사과정
<관심분야> RFID, 무선 네트워크 보안, 전자문서 보안

전 문 석 (Moon-seog Jun) 정회원
1981년 2월: 숭실대학교 전자계산학과 졸업
1986년 2월: University of Maryland Computer Science 석사
1989년 2월: University of Maryland Computer Science 박사
1986년 9월~1989 12월: University of Maryland 강사
1989년 3월~7월: Morgan State University 조교수
1989년 9월~1991년 2월: New Mexico State University Physical Science Lab. 책임연구원
1991년 3월 ~ 현재: 숭실대학교 정교수
<관심분야> 정보보호, 네트워크 보안, 전자여권, 암호학