情報保護學會論文誌
第 22 卷 第 3 號, 2012. 6

서버가 없는 환경에서 안전한 RFID 다중 태그 검색 프로토콜*

이 재 동†‡
경남대학교

A Secure RFID Multi-Tag Search Protocol Without On-line Server*

Jae-Dong Lee†‡
Kyungnam University

요 약

여러 응용 분야에서 서버의 도움 없이 리더는 태그들의 그룹에 특정 태그가 존재하는지를 알 필요가 있다. 이것을 서버 없는 RFID 태그 검색(serverless RFID tag searching)이라 한다. 이를 위해 몇 개의 프로토콜이 제시되었다. 하지만 이들 프로토콜들은 한 번에 하나의 태그를 검색하는 단일 태그 검색 프로토콜이다. 본 논문에서는 해시함수와 난수 발생기에 기반하여 한 번에 여러 개의 태그를 검색할 수 있는 다중 태그 검색 프로토콜을 제안한다. 이를 위해, S3PR 프로토콜[1]의 문제점으로 지적된 통신 오류에 의한 시드의 동기화 문제를 해결하는 프로토콜을 제안하고, 이를 기반으로 통신량을 줄일 수 있는 다중 태그 검색 프로토콜을 제안한다. 제안된 프로토콜은 추적공격, 위장공격, 재생공격 및 서비스 거부 공격에 안전하다. 이 연구는 다중 태그 검색 프로토콜 개발의 기초가 될 것이다.

ABSTRACT

In many applications a reader needs to determine whether a particular tag exists within a group of tags without a server. This is referred to as serverless RFID tag searching. A few protocols for the serverless RFID searching are proposed but they are the single tag search protocol which can search a tag at one time. In this paper, we propose a multi-tag search protocol based on a hash function and a random number generator which can search some tags at one time. For this study, we introduce a protocol which can resolve the problem of synchronization of seeds when communication error occurs in the S3PR protocol[1], and propose a multi-tag search protocol which can reduce the communication overhead. The proposed protocol is secure against tracking attack, impersonation attack, replay attack and denial-of-service attack. This study will be the basis of research for multi-tag serach protocol.

Keywords: RFID, security, serverless RFID search protocol, multi-tag search protocol

I. 서 론

RFID(Radio Frequency IDentification) 시스템은 유비쿼터스 컴퓨팅의 중요한 기반 기술로 자리잡고 있다. 이것은 RFID 태그(tag)들을 사용하여 사물(또는 사람)에 정보를 저장하고 사물과 떨어진 곳에서 리더(reader)를 사용하여 사물의 정보를 접근할 수 있는 자동 인식 시스템이다. 사물에 부착된 태

접수일(2011년 6월 8일), 수정일(1차: 2011년 8월 24일, 2차: 2011년 10월 5일), 게재확정일(2011년 10월 13일)
* 이 연구결과물은 2011학년도 경남대학교 학술연구장려금 지원에 의한 것임.
† 주저자, jdlee@kyungnam.ac.kr
‡ 교신저자, jdlee@kyungnam.ac.kr

그들은 인증 정보를 가지고 있으며, 이것을 사용하여 재고관리, 물품관리, 물류 및 유통, 고용자 관리, 미아 찾기 등 많은 분야에 활용되고 있으며, 미래에는 더 많은 분야에 활용될 것이다.

이런 응용 분야에서 RFID 시스템이 제공해야할 중요한 기능 중 하나는 '원하는 태그(들)가 주위에 존재하는가'를 탐지하는 태그 검색(Tag Searching)이다. 태그 검색의 필요성을 더 잘 이해하기 위해 아래에 2 개의 시나리오를 기술한다. (**시나리오1 : 항구에서 컨테이너 검색**) 보통, 항구에는 수백 또는 수천 개의 컨테이너가 적재되어 있다. 컨테이너들은 수많은 컨테이너 운전자들에 의해 운반되고 수많은 인부들에 의해 적재된다. 또한, 컨테이너들은 다른 곳으로 이동되기 위해 배에서 하역된다. 원하는 컨테이너(들)가 배에서 하역되었는지 또는 배에 선적하기 위해 항구에 도착했는지 등은 항구에서 이루어지는 중요한 작업들 중 하나이다. 그러나, 사람이 직접 원하는 컨테이너(들)를 찾는 것은 힘든 작업이다. 이를 위한 하나의 해결책은 컨테이너 식별을 위해 각 컨테이너에 RFID 태그를 부착시키고, 이동형 리더를 사용하여 원하는 컨테이너(들)가 존재하는지를 검색하는 방법이다. (**시나리오2 : 창고에서 물품 검색**) 태그가 부착된 물품들이 저장되어 있는 창고를 생각해 보자. 창고 관리자가 원하는 물품(들)이 창고에 있는지를 알고자 할 때, 관리자는 이동형 리더를 사용하여 원하는 물품(들)을 검색하는 질의를 보낸다. 태그들의 응답을 통해 원하는 물품들을 검색할 수 있다.

태그 검색은 태그 인증(authentication)의 확장 개념이다. 태그 검색을 위해 태그들의 그룹에 있는 모든 태그들에 대하여 인증 프로토콜을 사용하여 원하는 태그를 검색할 수 있다. 하지만, 이것은 비효율적이다. 따라서, 태그 검색을 위한 새로운 프로토콜이 필요하다. 최근에 이동 리더들의 공급이 활발해지고 그 응용 분야도 넓어지고 있다. 이동 리더는 중앙의 데이터베이스 즉, 서버와 항상 연결되어 있을 수 없다. 이동 리더에 저장되는 데이터는 서버로부터 초기화된 후, 서버 없이 사용되다가 연결이 가능한 경우(또는 필요한 경우)에 데이터를 업데이터 시킨다. 서버가 없는 환경에서 태그 검색은 프라이버시와 보안 위협에 노출되어 있다[2]. 예를 들어, 모든 물품에 태그가 부착된 물품창고에서 어떤 물품(들)을 검색하기 위해 질의 메시지를 브로드캐스트(broadcast)한다고 하자. 만약, 원하는 태그가 존재하면 그 태그는 리더에게 응답할 것이다. 공격자(adversary)는 전달되는 응답을 통해 해당 물품이 주위에 있음을 알 수 있게 된다. 이런 보안 위협은 태그 검색에 흔히 발생한다. 그래서 안전하고 실용적인 RFID 태그 검색 프로토콜 개발이 매우 중요한 연구 주제이다.

그러나, 서버 없는 환경에서 RFID 태그 검색에 대한 연구는 미미하다. 이 분야의 연구는 Tan 등이 발표한 논문[3]에서 처음으로 제시되었다. Tan 등이 제시한 프로토콜은 해시함수와 난수 발생기를 이용한다. 리더는 검색하려는 태그의 정보(태그의 식별자(id), 비밀키 등)의 해시함수의 값을 브로드캐스트 한다. 질의 메시지를 받은 태그는 이 메시지를 자신의 정보와 해시함수를 이용하여 검색하려는 태그인지를 확인한다. 검색된 태그는 자신의 정보를 이용한 해시값을 리더로 전송하고, 그 외의 태그들은 임의의 난수를 리더로 전송한다. 리더는 태그들로부터 받은 응답 메시지로부터 검색하려는 태그인지를 판별한다. Tan 등이 제시한 프로토콜처럼 해시함수와 난수발생기를 기반으로 한 연구로 Ahamed 등이 제시한 S^3PR 프로토콜[1]과 Zuo가 제시한 프로토콜[4]이 있다. 또한, 암호화 알고리즘을 기반으로 하는 프로토콜이 제시되었다[5, 6]. 그러나, 이들 연구들은 한 번에 하나의 태그를 검색하는 단일 태그 검색 프로토콜이다. 한 번에 여러 개의 태그를 검색하는 연구로 Zuo가 제시한 프로토콜이 있다. 그러나, 이 프로토콜은 서버를 필요로 하는 프로토콜이다. 이 프로토콜은 2 단계로 이루어져 있다. 첫 단계는 리더와 태그 사이의 통신을 통해 얻은 정보를 이용해 서버의 데이터베이스를 참조하여 리더와 태그 상이의 세션키를 생성한다. 두 번째 단계에서는 이 세션키를 사용하여 리더와 태그 사이의 통신이 이루어진다.

본 연구에서는 해시함수와 난수 발생기에 기반하여 한 번에 여러 개의 태그를 검색할 수 있는 프로토콜, 즉, 다중 태그 검색 프로토콜을 제안한다. 이를 위해, S^3PR 프로토콜의 문제점으로 지적된 통신 오류에 의한 시드의 동기화 문제[5, 6]를 해결하는 프로토콜을 제안하고, 이를 기반으로 통신량을 줄일 수 있는 다중 태그 검색 프로토콜을 제안한다. 제안된 프로토콜은 추적공격, 위장공격, 재생공격 및 서비스 거부 공격에 안전하다. 이 연구는 다중 태그 검색 프로토콜 개발의 기초가 될 것이다.

본 논문의 구성은 2장에서는 RFID 시스템 구조 및 위협 모델을 정의하고, 3장에서는 S^3PR 프로토콜 및 문제점을 기술하고, 4장에서는 S^3PR 프로토콜의 문제점을 해결한 단일 태그 검색 프로토콜을 제안한

다. 또한, 이것을 기반으로 하는 다중 태그 검색 프로토콜을 제안하고 분석한다. 5장에서는 결론 및 향후 연구 과제를 제시한다.

II. RFID 시스템 구조 및 위협 모델

2.1 RFID 시스템 구조

일반적으로 RFID 시스템은 보통 리더(reader), 태그(tag), 그리고 데이터베이스(또는 서버)로 구성된다. 리더와 태그 사이의 통신은 무선으로 이루어지고, 리더와 데이터베이스 사이의 통신은 무선 또는 유선으로 이루어진다. 태그는 리더의 요구에 자신의 id(identification number) 또는 태그에 저장된 정보들을 보내면, 이것을 받은 리더는 데이터베이스와 통신하여 태그와 관련된 일을 처리한다. 리더와 데이터베이스 간의 통신은 표준암호화 기술을 사용하므로 안전한 것으로 가정한다, 반면, 리더와 태그 사이의 모든 통신은 공격자가 도청가능하다고 가정한다. 그러나, 이 논문에서 다룰 RFID 시스템은 서버가 없는 시스템이므로 리더와 태그들도 구성되어 있다. 인증기관(certification authority) CA는 항상 신뢰되는 기관으로 태그들에 대한 정보들을 저장한 데이터베이스를 가지고 있어서 리더들을 인증하고 리더들이 태그들을 검색하기 위한 정보들을 제공하는 데만 관여한다. 이 시스템에서의 태그들은 가격이 저렴하고 자원(메모리, CPU등)이 제한적인 수동형 태그로 한정한다.

2.2 위협 모델(Threat Model)

RFID 태그의 제한적 자원(메모리, 프로세서의 성능)과 태그와 리더 사이의 무선 통신 때문에 RFID 시스템은 여러 가지 악의적인 공격에 노출 되어 있다. 여기에 기술되는 위협 모델을 위해, 공격자는 태그와 리더 간의 모든 통신을 가로챌 수 있으며 전송되는 메시지의 내용을 고칠 수 있다고 가정한다. 본 논문에서 고려할 태그 검색 과정에서의 태그 보안과 프라이버시에 대한 공격은 Kuo[4]가 제시한 모델을 사용하였다. 그 내용은 아래와 같다.

(1) 태그 위치 추적(tag location tracking) : 공격자가 태그와 리더 사이의 통신 내용을 도청하여 검색하고자 하는 태그를 추적하여 그 태그의 위치를 파악하는 공격.

(2) 태그 위장 공격(tag impersonation attack) : 공격자는 태그와 리더 사이의 통신 내용을 가로채서 통신 내용에 들어 있는 태그의 정보를 사용하여 정당한 태그로 위장하여 공격할 수 있다. 이 논문에서는 태그를 훔쳐 태그를 위장 공격하는 것은 제외한다.

(3) 재생 공격(replay attack) : 현재 검색 시, 공격자는 태그를 위장하기 위해 과거의 검색에서 사용된 태그들의 응답 메시지를 재사용하여 공격한다.

(4) 서비스 거부 공격(denial-of-service attack) : 공격자는 태그가 보내는 메시지를 가로 막거나 또는 태그의 작동을 일시적 또는 영구적으로 작동하지 못하게 하여 정상적인 검색 작업이 이루어지지 않게 한다.

III. S^{3}PR 프로토콜[1]

3.1 검색 프로토콜

3.1.1 표기법 및 가정

모든 리더들과 태그들은 난수발생기 $P(\cdot)$와 함수 $M(\cdot)$을 처리할 수 있다. $P(\cdot)$는 저비용으로 구현될 수 있는 간단한 난수발생기이며, 인자로 시드(seed)를 받아 하나의 난수를 생성한다. $M(\cdot)$은 현재의 시드를 받아 다음 시드를 구하는 한방향 해시함수(one way hash function)이다. 따라서 현재의 시드로부터 직전 시드를 유도하는 것이 불가능하다. 이렇게 생성된 시드는 다음의 난수를 구할 때 $P(\cdot)$에서 사용한다. RFID 리더를 R로 표현하고, 각 R은 자신의 식별자(identifier)인 r과 접근리스트 L을 가진다. R은 r과 L을 CA로부터 받는다. 각 태그 T는 자신의 식별자와 비밀키 t를 비휘발성 메모리에 저장하고 있다. 이 프로토콜에서 사용하는 표기법은 [표 1]과 같다.

첨자들은 R 또는 T와 그들과 관련된 파라메타들을 나타내기 위해 사용한다. 리더$i(R_i)$는 식별자 r_i와 접근리스트 L_i를 비휘발성 메모리에 저장하고 있으며, 태그 $j(T_j)$는 비밀키 t_j를 저장하고 있다. 접근리스트 L_i는 R_i가 접근할 수 있는 태그들의 정보를 가지고 있다. 각 태그에 대한 정보는 시드와 식별자로 구성된다. R_i가 태그 T_1, …, T_n을 접근할 수 있도록 허가되

〔표 1〕 프로토콜에서 사용하는 표기법

표기	의미
R_i	검색하려는 RFID 리더 i
$T_{desired}$	리더가 검색하려는 태그
$\text{seed}_{desired}$	R_i의 접근리스트에 있는 $T_{desired}$에 대한 시드
$seed_{T_{desired}}$	태그 $T_{desired}$에 저장되어 있는 시드
$n_{desired}$	리더 R_i가 태그 $T_{desired}$를 검색하기 위해 $seed_{desired}$로 생성한 난수
T^*	리더 R_i 주변에 있는 모든 태그들
$seed_{T^*}$	R_i 주변에 있는 모든 태그들에 저장되어 있는 시드
$\|\|$	연접(concatenation)

면 L_i는 아래와 같은 정보를 가진다.

$$L_i = \begin{Bmatrix} seed_1 : id_1 \\ seed_2 : id_2 \\ \cdots \\ seed_n : id_n \end{Bmatrix}$$

여기서, $seed_j$는 R_i가 태그 T_j와 통신하기 위한 시드를 나타내고 id_j는 T_j의 식별자를 나타낸다. $seed_j$의 초기값은 해시함수 $h(\cdot)$을 사용하여 아래와 같이 CA에서 계산되어 R_i에게 주어진다.

$$seed_j = f(r_i, t_j) = h(r_i \| t_j)$$

R_i는 태그의 비밀키 t_j을 알 수 없으며, 단지 시드 $seed_j$ 만을 가진다. 또한, 각 태그 T_j는 허가된 모든 리더에 대하여 각각의 시드를 저장하고 있다. T_j가 저장하고 있는 R_i에 대한 시드 $seed_{T_j}$는 이 태그가 CA에 의해 처음 사용되어질 때 CA로부터 아래와 같이 계산되어 태그에 저장된다.

$$seed_{T_j} = f(r_i, t_j) = h(r_i \| t_j)$$

3.1.2 검색 프로토콜

제시한 검색 프로토콜은 〔그림 1〕과 같다. 먼저, 검색을 수행할 리더 R_i는 검색할 태그 $T_{desired}$의 시드 $seed^k_{desired}$를 이용하여 난수 $n^k_{desired}$를 계산한다. 이것을 브로드캐스트한다. 이것을 받은 태그들은 그들 자신이 저장하고 있는 R_i의 시드 $seed^k_i$를 사용하여 만든 난수 n^k_i와 $n^k_{desired}$를 비교한다. 같으면, 다음 번 시드 $seed^{k+1}_i$를 계산하고, 이를 사용하여 난수 n^{k+1}_i을 구한다. 이를 리더에 보내고 태그의 시드를 다음 번 시드인 $seed^{k+2}_i$로 바꾸어 저장한다. 같지 않는 태그들은 확률 λ에 따라 임의의 난수를 생성하여 리더에 보낸다.

태그들로부터 난수들을 받은 리더 R_i는 리더에 저장되어 있는 $seed^k_{desired}$를 사용하여 다음 난수 $seed^{k+1}_{desired}$를 생성하고, 이를 사용하여 난수 $n^{k+1}_{desired}$을 생성한다. 태그들로부터 받은 난수들 중 $n^{k+1}_{desired}$과 일치하는 태그가 있으면 이 태그가 검색하고자 하는 태그가 된다. 검색하려는 태그를 찾았으면 그 태그에 대한 시드를 다음 시드 값, 즉, $seed^{k+2}_{desired}$로 재설정한다.

3.1.3 보안 분석

이 프로토콜은 아래에 설명한 추적공격(tracking), 복제공격(cloning), 및 도청(eavesdropping)에 안전하다〔1〕.

- 추적공격 : 공격자(adversary)가 검색하려는 태그를 추적하여 해당 태그를 찾는 공격
- 복제공격 : 공격자가 태그를 복제하여 리더의 검색을 혼란시키는 공격
- 도청 : 리더와 태그들 간의 통신을 도청을 통한 공격

3.2 프로토콜의 문제점

이 프로토콜에는 아래와 같은 문제점이 발생하다〔5, 6〕.

(1) 다수의 리더를 사용할 때 리더들 간에 태그들의 시드에 대한 동기화가 필요하다.

(2) 리더와 태그들 간의 통신에 오류〔7〕가 발생하거나 또는 서비스 거부 공격에 의하여 탐색된 태그에서 리더로 보낸 n^{k+1}_i을 리더가 받지 못했을 때, 태그의 시드는 $seed^{k+2}_i$로 수정되었지만 리더에서는 해당 태그의 시드가 수정되지 않은 문제가 발생한다.

IV. 다중 태그 검색 프로토콜

4.1절에서는 3.2절에서 지적한 문제점들을 해결하는 방안을 제시하고, 4.2절에서는 해결 방안을 토대로 다중 태그 검색 프로토콜을 제시한다.

(1) R_i : Commpute $seed^k_{desired} = seed_{desired}$, $n^k_{desired} = P(seed^k_{desiredi})$
(2) $R_i \rightarrow T^*$: *Broadcast* $n^k_{desired}$
(3) T^* : $seed^k_i = seed_i$, $n^k_i = P(seed^k_i)$
(4) *If* $(n^k_i = n^k_{desired})$ *then*
(5) $n^{k+1}_i = P(seed^{k+1}_i)$
(6) $seed_i = seed^{k+2}_i$
(7) $R_i \leftarrow T_j : n^{k+1}_i$
(8) *Else*
(9) $R_i \leftarrow T_j$: *a random number with probablity* λ
(10) R_i : $n^{k+1}_{desired} = P(seed^{k+1}_{desired})$
(11) *If* $(n^{k+1}_i = n^{k+1}_{desired})$ *then*
(12) $seed_{desired} = seed^{k+2}_{desired}$
(13) $T_{desired}$ *found*
(14) *Else*
(15) $T_{desired}$ *not found*

〔그림 1〕 검색 프로토콜

4.1 수정 프로토콜

3.2절에서 지적한 (1)번의 문제점은 프로토콜 기술의 모호함과 이 프로토콜을 잘못 이해한데서 기인한다. 이를 위해 태그에 저장되어 있는 시드들의 값을 정확히 정의하면 〔그림 2〕와 같다.

리더번호	시드 값
1	$seed_1$
2	$seed_2$
3	$seed_3$
4	$seed_4$
.	.
.	.
.	.
.	.

〔그림 2〕 각 태그에 저장된 시드

〔그림 2〕처럼 각 태그에는 리R_i더 별로 다른 시드 값을 저장하고 있다. 따라서, 리더에 저장된 $seed_j$와 태그T_j의 $seed_i$가 같게 유지되면 되므로 리더들 간의 태그들의 시드에 대한 동기화는 필요 없다. 또한, 프로토콜의 모호함을 없애기 위해 각 태그들은 브로드캐스팅한 값을 받았을 때, 어떤 리더가 보냈는지를 구별할 수 있다고 가정하던가 아니면 프로토콜의 (2)번 라인을 아래와 같이 수정하여 검색하는 리더의 번호를 명확히 하면 된다.

$$R_i \rightarrow T^* : \quad Broadcast \quad i, \quad n^k_{desired} \qquad (2)$$

(2)번의 문제점은 태그에서 보낸 내용을 리더가 받지 못하여 리더와 태그의 시드가 일치하지 않아서 생기는 문제이다. 이를 해결하기 위해 〔4〕에서 사용한 방법을 이용할 수 있다. 하지만 이 방법은 2번 이상의 연속된 통신 에러가 발생했을 경우 리더와 태그의 시드를 동기화 할 수 없을 뿐만 아니라 추적공격에도 취약하다. (2)번째 문제점을 해결하기 위해 〔그림 1〕을 수정한 검색 프로토콜을 제시한다. 각 태그들은 각 리더별로 두 개의 시드를 저장한다. 즉, 〔그림 2〕를 수정하여 〔그림 3〕처럼 유지시킨다. 초기의 $seed^{old}_i$와 $seed^{new}_i$는 동일한 값이다. $seed^{old}_i$는 검색이 되었을 때 바뀌기 전 시드, 즉, $seed^k_i$이고, $seed^{new}_i$는 바뀐 시드, 즉 $seed^{k+2}_i$이다. 수정한 프로토콜은 〔그림 4〕와 같다.

리더번호	시드 값	
1	$seed^{old}_1$	$seed^{new}_1$
2	$seed^{old}_2$	$seed^{new}_2$
3	$seed^{old}_3$	$seed^{new}_3$
4	$seed^{old}_4$	$seed^{new}_4$
.	.	.
.	.	.
.	.	.

〔그림 3〕 수정된 검색 프로토콜에서 사용될 태그에 저장된 시드

(1) R_i : *Compute* $seed^k_{desired} = seed_{desired}$, $n^k_{desired} = P(seed^k_{desiredi})$
(2) $R_i \rightarrow T^*$: *Broadcast* i , $n^k_{desired}$
(3) T^* : $n_i^{old} = P(seed_i^k)$, $n_i^{new} = P(seed_i^{new})$
(4) *If* $(n^k_{desired} = n_i^{old}$ or $n^k_{desired} = n_i^{new})$ *then*
(5) *If* $(n^k_{desired} = n_i^{old})$ $seed_i^k = seed_i^{old}$
(6) *Else* $seed_i^k = seed_i^{new}$
(7) $seed_i^{k+1} = M(seed_i^k)$, $seed_i^{k+2} = M(seed_i^{k+1})$
(8) $n_i^{k+1} = P(seed_i^{k+1})$
(9) $seed_i^{old} = seed_i^k$, $seed_i^{new} = seed_i^{k+2}$
(10) $R_i \leftarrow T_j$: n_i^{k+1}
(11) *Else*
(12) $R_i \leftarrow T_j$: *a random number with probablity* λ
(13) R_i : $n^{k+1}_{desired} = P(seed^{k+1}_{desired})$
(14) *If* $(n_i^{k+1} = n^{k+1}_{desired})$ *then*
(15) $seed_{desired} = seed^{k+2}_{desired}$
(16) $T_{desired}$ *found*
(17) *Else*
(18) $T_{desired}$ *not found*

〔그림 4〕 수정된 검색 프로토콜

(1) R_i : *Compute* $seed^k_{desired} = seed_{desired}$, $n^k_{desired} = P(seed^k_{desiredi})$
(2) $R_i \rightarrow T^*$: *Broadcast* i , $n^k_{desired}$
(3) T^* : $n_i^{old} = P(seed_i^k)$, $n_i^{new} = P(seed_i^{new})$, n_t = *a random number*
(4) *If* $(n^k_{desired} = n_i^{old}$ or $n^k_{desired} = n_i^{new})$ *then*
(5) *If* $(n^k_{desired} = n_i^{old})$ $seed_i^k = seed_i^{old}$
(6) *Else* $seed_i^k = seed_i^{new}$
(7) $seed_i^{k+1} = M(seed_i^k)$, $seed_i^{k+2} = M(seed_i^{k+1})$
(8) $n_i^{k+1} = P(seed_i^{k+1} \oplus n_t)$
(9) $seed_i^{old} = seed_i^k$, $seed_i^{new} = seed_i^{k+2}$
(10) $R_i \leftarrow T_j$: n_i^{k+1}, n_t
(11) *Else*
(12) $R_i \leftarrow T_j$: two *random numbers with probablity* λ
(13) R_i : $n^{k+1}_{desired} = P(seed^{k+1}_{desired} \oplus n_t)$
(14) *If* $(n_i^{k+1} = n^{k+1}_{desired})$ *then*
(15) $seed_{desired} = seed^{k+2}_{desired}$
(16) $T_{desired}$ *found*
(17) *Else*
(18) $T_{desired}$ *not found*

〔그림 5〕 단일 태그 검색 프로토콜

이를 수정된 검색 프로토콜이라 부르겠다.

수정된 검색 프로토콜에서는 기존 프로토콜처럼 탐색하려는 태그의 시드 $seed^k_{desired}$를 사용하여 $n^k_{desired}$를 생성하여 브로드캐스트 한다. 각 태그들은 $seed_i^{old}$와 $seed_i^{new}$를 사용하여 n_i^{old} 또는 n_i^{new}를 생성하고 $n^k_{desired}$와 비교한다. n_i^{old} 또는 n_i^{new}와 일치하면 탐색하려는 태그이고, 그렇지 않으면 탐색하려는 태그가 아니다. 탐색하려는 태그이면 $n^k_{desired}$가 n_i^{old} 인가 또

는 n_i^{new} 인가에 따라 $seed_i^k$의 값을 $seed_i^{old}$ 또는 $seed_i^{new}$로 설정한 다음, 기존 프로토콜과 같이 n_i^{k+1}을 생성하여 리더로 전송하고, $seed_i^{old}$는 $seed_i^k$로, $seed_i^{new}$는 $seed_i^{k+2}$로 변경하여 저장한다. 탐색하려는 태그가 아니면 임의의 난수를 생성하여 리더로 전송한다.

이 수정된 검색 프로토콜은 전송 상의 에러가 발생했을 경우, 리더와 태그간의 시드의 동기화를 이룰 수 있다. 하지만, 공격자가 리더에서 브로드캐스팅 한 $i, n_{desired}^k$를 도청하여 공격자가 리더로 가장하여 이 값을 브로드캐스트 하면 검색하려는 태그는 계속 같은 내용을 보내는 반면, 다른 태그들은 임의의 난수를 보내므로 검색하려는 태그를 추적할 수 있다. 이를 해결하기 위해 수정된 검색 프로토콜을 약간 수정한 〔그림 5〕와 같은 단일 태그 검색 프로토콜을 제시한다.

단일 태그 검색 프로토콜에서는 앞에서 지적한 공격을 막기 위해 검색된 태그는 임의의 난수를 하나 생성하여 이 난수 n_t를 사용하여 $n_i^{k+1} = P(seed_i^{k+1} \oplus n_t)$ 을 생성하고, n_i^{k+1}와 n_t을 리더로 보낸다. 리더는 전송받은 n_t를 사용하여 $n_{desired}^{k+1} = P(seed_{desired}^{k+1} \oplus n_t)$를 계산하여 전송받은 n_i^{k+1}와 비교한다. 검색되지 않은 태그들은 응답하는 메시지의 크기를 같게 하기 위해 임의의 두 난수를 생성하여 리더로 전송한다. 이렇게 함으로써 앞에서 지적한 공격을 막을 수 있다.

4.1.1 보안 분석

단일 태그 검색 프로토콜에 대한 보안은 아래와 같이 태그 위치 추적, 태그 위장 공격, 재생 공격 및 서비스 거부 공격에 안전하다.

- 태그 위치 추적 : 공격자가 질의 메시지와 응답 메시지를 도청하면 공격자는 i번 리더가 하나의 태그를 검색하려고 한다는 것은 알 수 있다. 하지만 많은 태그들이 응답메시지로 난수를 전송하므로 이 난수들을 풀지 않는 한 검색된 태그들이 어떤 것인지는 추적할 수 없다. 공격자는 시드를 알 수 없으므로 태그들이 전송하는 난수를 풀 수 없으며 또한, 리더가 보내는 질의 메시지도 풀 수 없다.
- 태그 위장 공격 : 태그와 리더 간의 통신 메시지에는 단지 난수들만 존재한다. 이 난수들로부터 태그의 정보를 얻어 태그를 위장하는 것은 불가능하다. 다만, 태그를 훔쳐 위장하는 것은 가능하나 이것은 이 논문의 범위를 벗어 난다.
- 재생 공격 : 공격자가 질의 메시지를 도청하여 자신이 리더인 것처럼 가장하여 그 메시지를 태그들에게 브로드캐스트하여 검색하려는 태그들을 추적한다고 하자. 태그들의 응답 메시지는 매 검색마다 다른 난수를 보내기 때문에 이런 공격으로도 검색하려는 태그들을 추적할 수 없다. 또한, 태그의 응답 메시지를 도청해 두었다가 다음 검색에 사용한다고 가정하자. 이때, 도청해둔 응답메시지가 검색하려는 태그의 응답인지 알 수 없다. 따라서 이런 재생 공격도 불가능하다.
- 서비스 거부 공격 : 어떤 태그의 응답 메시지가 전송되는 것을 막아 검색을 못하게 한다고 가정하자. 이 때, 검색하고자 하는 태그가 어느 것인지 알 수 없으므로 검색을 막을 수 없다. 운 좋게 검색하려는 태그의 응답메시지의 전송을 막았다고 하자. 하지만, 이것으로 다음 검색을 막을 수는 없다. 검색하려던 태그의 응답메시지가 전달되지 않으면 태그는 시드를 바꾸지만 리더의 시드는 변경되지 않는다. 하지만 이 문제를 해결하기 위해 태그에 이전 시드와 바뀐 시드를 유지하므로 다음 검색에는 문제가 없다.
- 통신 오류 : 검색된 태그의 응답메시지가 통신 장애로 전달이 안 되었을 경우, 위에서 언급한 서비스 거부 공격과 마찬가지로 태그에 이전 시드와 바뀐 시드를 유지함으로써 다음 검색에는 문제가 없다.

4.2 다중 태그 검색 프로토콜

한 번에 여러 개의 태그들을 검색하는 것을 다중 태그 검색이라고 한다. 한 번에 찾을 태그들의 개수를 m이라 하면 리더는 m개의 난수를 한꺼번에 전송하여야 한다. 이를 위해 4.1절에서 제안한 단일 태그 검색 프로토콜을 이용하여 두 개의 프로토콜을 제안한다.

4.2.1 기본 다중 태그 검색 프로토콜

검색하려는 m개의 태그들에 대한 난수들을 $n_{desired1}$, $n_{desired2}$, … , $n_{desiredm}$이라 하자. $n_{desiredi}$는 $P(seed_{desiredi}^k)$로 계산할 수 있다. 기본 다중 태그 검색 프로토콜은 〔그림 6〕과 같다.

기본 다중 태그 검색 프로토콜에서는 검색하려는

태그들의 난수들, 즉, $n^k_{desired1}, n^k_{desired2}, \cdots, n^k_{desiredm}$를 계산하여 이들을 브로드캐스트 한다. 이들을 전송받은 태그들은 단일 태그 검색 프로토콜처럼 n_i^{old}와 n_i^{new}를 계산하여 자신이 검색하려는 태그인지를 확인하기 위해 $n^k_{desiredj}(j=1,\cdots,m)$들과 비교한다. 자신이 검색하려는 태그이면 단일 태그 검색 프로토콜처럼 시드를 수정하고, 임의의 난수 n_t와 $seed_i^{k+1}$를 이용하여 $n_i^{k+1} = P(seed_i^{k+1} \oplus n_t)$ 을 계산하여 n_i^{k+1}와 n_t를 리더로 보낸다. 검색하려는 태그가 아니면 단일 태그 검색 프로토콜처럼 두 개의 임의의 난수를 생성하여 리더로 보낸다. 리더는 전송 받은 n_i^{k+1}과 n_t를 사용하여, $j = 1, \cdots, m$에 대하여 $n^{k+1}_{desiredj} = P(seed^{k+1}_{desiredj} \oplus n_t)$를 계산하여 n_i^{k+1}와 비교한다. 매칭이 발생하면 그 태그는 검색하려는 태그이고, 그렇지 않으면 검색하려는 태그가 아니다. 검색하려는 태그이면 시드를 수정한다.

4.2.2 다중 태그 검색 프로토콜

기본 다중 태그 검색 프로토콜에서는 리더가 태그들에게 브로드캐스트 하는 데이터가 너무 크다. 이 전송 데이터의 크기를 하나의 태그만을 검색하는 단일태그 검색 프로토콜에서 브로드캐스트 하는 정도의 크기로 줄여 다중태그를 검색하는 프로토콜을 제시한다.

리더는 검색할 각 태그 $desired_j$에 대한 난수 $n^k_{desiredj}$를 구하고, mod 연산을 이용하여 $D^k_{desiredj}$를 아래와 같이 구한다. 여기서 b는 난수의 비트 수이다. $D^k_{desiredj}$는 b비트의 난수 중 하위 $\lfloor \frac{b}{m} \rfloor$ 비트를 나타낸다.

$$D^k_{desiredj} = n^k_{desiredj} \bmod 2^{\lfloor \frac{b}{m} \rfloor},$$

m은 동시에 검색할 태그들의 개수

이 프로토콜에서는 기본 다중 태그 검색 프로토콜에서 브로드캐스트 한 $n^k_{desired1}, n^k_{desired2}, \cdots, n^k_{desiredm}$ 대신 $D^k_{desired1}, D^k_{desired2}, \cdots, D^k_{desiredm}$를 브로드캐스트 한다. 각 태그들은 이 값들을 이용하여 검색 여부를 판별하며 기본 다중 태그 검색 프로토콜과 똑같이 시드를 수정하고 n_i^{k+1}과 n_t를 리더로 보낸다. 리더는 태그로부터 받은 n_i^{k+1}을 기본 다중 태그 검색 프로토

```
(1) Ri              : Compute n^k_desired1, n^k_desired2, ..., n^k_desiredm
(2) Ri → T*         : Broadcast i, m, n^k_desired1, n^k_desired2, ..., n^k_desiredm
(3) T*              : n_i^old = P(seed_i^old), n_i^new = P(seed_i^new), n_t = a random number
(4)                   F = 0
(5)                   For j = 1, ... , m  do
(6)                         If (n^k_desiredj = n_i^old or n^k_desiredj = n_i^new) F = j,  Break
(7)                   If (F = 0)
(8)                         Ri ← Tj : two random numbers with probability λ
(9)                   Else
(10)                        If (n^k_desiredF = n_i^old) seed_i^k = seed_i^old
(11)                        Else seed_i^k = seed_i^new
(12)                        seed_i^{k+1} = M(seed_i^k), seed_i^{k+2} = M(seed_i^{k+1})
(13)                        n_i^{k+1} = P(seed_i^{k+1} ⊕ n_t)
(14)                        seed_i^old = seed_i^k, seed_i^new = seed_i^{k+2}
(15)                        Ri ← Tj : n_i^{k+1}, n_t
(16) Ri             : F = 0
(17)                  For j = 1, ... , m  do
(18)                        Compute n^{k+1}_desiredj = P(seed^{k+1}_desiredj ⊕ n_t)
(19)                        If (n_i^{k+1} = n^{k+1}_desiredj)  F = j ,  Break
(20)                  If (F != 0) then
(21)                        seed_desiredF = seed^{k+2}_desiredF
(22)                        T_desiredF found
```

〔그림 6〕 기본 다중 태그 검색 프로토콜

(1) R_i : Compute $D^k_{desired1}$, $D^k_{desired2}$, ⋯, $D^k_{desiredm}$
(2) $R_i \rightarrow T^*$: Broadcast i, m, $D^k_{desired1}$, $D^k_{desired2}$, ⋯, $D^k_{desiredm}$
(3) T^* : $n_i^{old} = P(seed_i^{old})$, $n_i^{new} = P(seed_i^{new})$, n_t = a random number
(4) $D_i^{old} = n_i^{old} mod\ m$, $D_i^{new} = n_i^{new} mod\ m$
(5) $F = 0$
(6) For $j = 1, \cdots, m$ do
(7) If ($D^k_{desiredj} = D_i^{old}$ or $D^k_{desiredj} = D_i^{new}$) $F = j$, Break
(8) If ($F = 0$)
(9) $R_i \leftarrow T_j$: two random numbers with probability λ
(10) Else
(11) If ($D^k_{desiredF} = D_i^{old}$) $seed_i^k = seed_i^{old}$
(12) Else $seed_i^k = seed_i^{new}$
(13) $seed_i^{k+1} = M(seed_i^k)$, $seed_i^{k+2} = M(seed_i^{k+1})$
(14) $n_i^{k+1} = P(seed_i^{k+1} \oplus n_t)$
(15) $seed_i^{old} = seed_i^k$, $seed_i^{new} = seed_i^{k+2}$
(16) $R_i \leftarrow T_j$: n_i^{k+1}, n_t
(17) R_i : $F = 0$
(18) For $j = 1, \cdots, m$ do
(19) Compute $n^{k+1}_{desiredj} = P(seed^{k+1}_{desiredj} \oplus n_t)$
(20) If ($n_i^{k+1} = n^{k+1}_{desiredj}$) $F = j$, Break
(21) If (F != 0) then
(22) $seed_{desiredF} = seed^{k+2}_{desiredF}$
(23) $T_{desiredF}$ found

〔그림 7〕 다중 태그 검색 프로토콜

콜과 같은 방법으로 수행하여 검색하려는 태그인지를 확인하고 시드를 수정한다. 프로토콜은 〔그림 7〕과 같다.

4.2.3 보안 및 비용 분석

다중 태그 검색 프로토콜은 〔그림 5〕의 단일 태그 검색 프로토콜을 기반으로 하고 있다. 두 프로토콜의 차이점은 단일 검색 프로토콜에서는 검색할 태그의 시드로 생성한 난수를 브로드캐스팅 하는 반면, 다중 태그 검색 프로토콜에서는 검색할 m 개의 태그들에 대하여 각 태그의 시드로 생성된 난수의 하위 $\lfloor \frac{b}{m} \rfloor$ 비트를 m개 모아 브로트캐스팅 한다. 리더로부터 수신한 각 태그들은 검색 결과가 포함된 난수를 리더로 송신하는 것은 두 프로토콜이 동일하다. 리더는 이 난수를 이용하여 검색을 마무리한다. 두 프로토콜에서 태그와 리더에 저장되어 있는 정보는 동일하며, 전송되는 정보는 위에서 언급한 검색 초기 단계에서 브로트캐스팅 하는 내용만 다르다. 단일 태그 검색 프로토콜에서는 난수를, 다중 태그 검색 프로토콜에서는 m 개의 난수들의 하위 $\lfloor \frac{b}{m} \rfloor$ 비트를 전송한다.

따라서, 다중 태그 검색 프로토콜의 보안은 단일 태그 검색 프로토콜의 보안과 동일하다. 즉, 4.1.1절에 기술한 것처럼 위치 추적 공격, 태그 위장 공격, 재생 공격 및 서비스 거부 공격에 안전하며, 통신 오류가 발생하더라도 다음 검색에는 아무런 문제가 발생하지 않는다.

두 프로토콜의 처리량과 통신량을 비교 분석하기 위해, 단일 태그 검색 프로토콜로 m번의 검색과 m개의 태그를 한 번에 검색하는 다중 태그 검색 프로토콜에서 리더와 태그의 처리량 및 통신량을 비교 분석하였다. 난수 n^k의 크기를 b 비트라 하고, 리더의 번호 i와 검색 태그 수 m은 b에 비하여 아주 작은 값이라 무시한다. 먼저, 처리량을 알아 본다. 단일 태그 검색 프로토콜로 1번의 검색을 위해서 태그에서는 상수 시간의 처리가, 리더에서는 수신되는 모든 태그들에 대해 처리해야 함으로 $O(\lambda n)$의 처리가 필요하다. 반면, 다중 태그 검색 프로토콜로 1번의 검색을 위해서, 태그는 〔그림 7〕의 (6)번 ~ (7)번 라인에서 $O(m)$의 처리가, 리더에서는 수신되는 모든 태그들에 대해 〔그림 7〕의 (18)번 ~ (20)번 라인에서 $O(m)$의 처리가

〔표 2〕 처리량과 통신량의 비교 (n : 태그의 개수)

	처리량	통신량(비트)
m번의 단일 태그 검색 프로토콜	태그 : $O(m)$ 리더 : $O(mn)$	$mb(1+2\lambda n)$
1번의 다중 태그 검색 프로토콜	태그 : $O(m)$ 리더 : $O(mn)$	$b(1+2\lambda n)$

필요하므로 전체적으로 $O(\lambda mn)$의 처리가 필요하다. 따라서, m번의 단일 태그 검색 프로토콜과 1번의 다중 태그 검색 프로토콜의 처리량은 리더에서 $O(mn)$, 태그에서 $O(m)$으로 같다.

다음으로 통신량을 비교한다. 단일 태그 검색 프로토콜로 1번의 검색을 할 경우, 리더에서 태그로 b 비트를, λn개의 태그가 리더로 각각 $2b$ 비트씩 보내므로 $b(1+2\lambda n)$의 통신량이 필요하다. 반면, m개의 태그를 한 번에 검색하는 다중 태그 검색 프로토콜을 사용하면, 리더에서 태그로 b 비트를, λn개의 태그가 리더로 각각 $2b$ 비트씩 보내므로 $b(1+2\lambda n)$의 통신량이 필요하다. 따라서, m번의 단일 태그 검색 프로토콜을 사용할 경우 통신량은 $bm(1+2\lambda n)$인 반면, 1번의 다중 태그 검색 프로토콜의 통신량은 $b(1+2\lambda n)$로 $1/m$ 만큼 적은 통신량으로 검색이 가능하다. 처리량과 통신량의 비교 분석 결과는 〔표 2〕와 같다.

4.2.4 기타 사항

다중 태그 검색 프로토콜을 사용하는 리더를 도난당한 경우를 생각해 보자. 다른 시스템과 마찬가지로 이 리더로 모든 태그들에 대한 검색이 가능하다. 하지만, 리더에 저장되어 있는 태그들의 시드를 이용하여 태그들을 복제(또는 위장)했을 경우, 도난당한 리더로부터의 검색은 가능하지만 도난당하지 않은 다른 리더로부터의 검색에 대해서는 올바른 응답을 할 수 없다.

우리가 제안한 다중 태그 검색 프로토콜은 해시함수와 난수발생기를 기반으로 하고 있다. 만약, 암호화 알고리즘을 기반으로 하는 다중 태그 검색 프로토콜을 생각해보자. m개의 태그를 검색하기 위해 리더는 각 태그의 정보를 그 태그의 비밀키로 암호화한 m개의 데이터를 기본 다중 태그 프로토콜처럼 브로드캐스트 한다. 이것을 받은 태그들은 자신의 비밀키로 복호화 하여 자신의 정보가 있는지를 판단한 후, 응답 메시지를 보내게 된다. 이 방법을 확장하여 통신량을 1/m로 줄인 다중 태그 검색 프로토콜과 같은 방법으로 할 수 있을까? 태그의 정보를 암호화한 데이터의 일부만 보내게 되면 태그에서는 제대로된 정보를 얻을 수 없다. 따라서, 암호화 알고리즘을 기반으로 우리가 제시한 다중태그 검색 프로토콜 방식을 만들기는 어려울 것이다.

우리가 제안한 다중 태그 검색 프로토콜을 다음과 같은 문제점을 안고 있다. 첫째, 리더의 개수가 r개라 할 때 각 태그에 $2r$ 개의 시드를 저장해야 된다. 둘째, 새로운 리더가 추가될 때, 각 태그에 새로운 리더에 대한 시드를 저장해야 함으로 이는 불가능하다. 따라서, 제안된 프로토콜은 리더의 개수가 작고 새로운 리더가 추가되지 않는 환경에 적합하다. 이 문제점들의 해결책은 향후 연구로 남겨 둔다.

V. 결론

유비쿼터스 컴퓨팅의 중요한 기반 기술의 하나인 RFID(Radio Frequency IDentification) 시스템에서 리더는 태그들의 그룹에 특정 태그가 존재하는지를 알 필요가 있다. 이것을 RFID 검색이라 한다. 이동 리더의 공급이 활발해지고 그 응용 분야도 넓어지고 있다. 이런 환경에서 서버의 도움 없이 리더는 태그의 검색이 필요하다. 그래서 안전하고 실용적인 서버 없는 환경에서의 RFID 검색 프로토콜 개발이 매우 중요한 연구 주제이다. 그러나, 서버 없는 환경에서 RFID 검색에 대한 연구는 미미한 수준이며, 대부분의 연구가 한 번에 하나의 태그를 검색하는 프로토콜에 제한되어 있다.

본 논문에서는 해시함수와 난수 발생기에 기반하여 한 번에 여러 개의 태그를 검색할 수 있는 프로토콜, 즉, 다중 태그 검색 프로토콜을 제안하였다. 이를 위해, S^3PR 프로토콜의 문제점으로 지적된 통신 오류에 의한 시드의 동기화 문제를 해결하는 단일 태그 검색 프로토콜을 제안하였고, 이를 기반으로 통신량을 줄일 수 있는 다중 태그 검색 프로토콜을 제안하였다. 제안된 다중 태그 검색 프로토콜은 m번의 단일 태그 검색 프로토콜에 비해 리더와 태그 사이의 통신량을 1/m 만큼 줄일 수 있고, 추적공격, 위장공격, 재생공격 및 서비스 거부 공격에 안전하다. 이 연구는 다중 태그 검색 프로토콜 개발의 기초가 될 것이라 생각한다.

향후, 제시한 다중 태그 검색 프로토콜의 문제점인 태그에 저장되는 시드의 개수를 줄이는 방안 및 새로운 리더의 추가가 가능한 프로토콜 연구가 필요하다.

참고문헌

〔1〕 S.I. Ahamed, F. Rahman, E. Hoque, F. Kawsar, and T. Nakajima, "S^3PR: Secure Serverless Search Protocols for RFID," *Proc. of the ISA08*, pp. 187-192, Apr. 2008.

〔2〕 A. Juels, "RFID Security and Privacy : A Research Survey," *RSA Laboratories*, Sep. 2005

〔3〕 C.C. Tan, B. Sheng, and Q. Li, "Secure and Serverless RFID Authentication and Search Protocols," *IEEE Transactions on Wireless Communications*, vol. 7, no. 3, pp. 1400-1407, Apr. 2008.

〔4〕 Yanjun Zuo, "Secure and private Search protocols for RFID systems," *Information Systems Frontiers*, vol. 12, pp. 507-519, Nov. 2010.

〔5〕 천지영, 황정연, 이동훈, "이동형 리더 소지자의 프라이버시를 보호하는 RFID 태그 검색 프로토콜," *정보보호학회논문지*, 19(5), pp. 59-69, 2009년 10월.

〔6〕 임지환, 오희국, 김상진, "온라인 서버가 없는 환경에서 이동형 리더의 프라이버시를 보호하는 안전한 RFID 검색프로토콜," *정보보호학회논문지*, 20(2), pp. 73-90, 2010년 4월.

〔7〕 P. Hernandez, J.D. Sandoval, F. Puente, and F. Perez, "Mathematical Model For a Multiread Anticollision Protocol," *IEEE Pacific Rim Conference on Communication, Computers and Signal Processing(PACRIM02)*, vol. 2, pp. 647-650, Aug. 2002.

〈著者紹介〉

이 재 동(Jae-Dong Lee) 정회원
1983년 2월: 서울대학교 계산통계학과 이학사
1985년 2월: 서울대학교 전산과학전공 이학석사
1995년 2월: 서울대학교 전산과학전공 이학박사
1986년~현재: 경남대학교 컴퓨터공학부 교수
〈관심분야〉 실시간 시스템, 임베디드 시스템, 컴퓨터 보안