RFID 보안 에이전트

박혜영°, 김성훈 , 박창윤
중앙대학교 컴퓨터네트워크연구실
best117@cnlab.cse.cau.ac.kr°, shkim@cnlab.cse.cau.ac.kr, cypark@cau.ac.kr

RFID Security Agent

Hye Young Park°, Sung Hun Kim, Chang Yun Park
Computer Network Lab., Chung-Ang University

요 약

본 논문에서는 현 RFID 시스템에 도입이 가능하고 저등급의 태그도 보호할 수 있는 보안 에이전트를 제안한다. 보안 에이전트는 태그를 대신하여 보안 모듈을 수행하는 구성요소로서 등록된 태그만을 대상으로 이 작업을 수행하게 되는데 보안 에이전트의 보호 범위에는 보호 대상인 태그와 보호대상이 아닌 태그가 존재하기 때문에, 이를 구별하여 보안 모듈을 수행할 수 있는 알고리즘이 필요하다. 보안 모듈 중 가장 간단한 블로킹 기법을 채택하여, 선택적으로 보호 대상인 태그만을 블로킹하는 알고리즘을 설명한다. 보안 에이전트는 추가 구성요소로 동작하기 때문에 현재 RFID 시스템의 변경 없이 도입 가능하여 초기 도입 비용이 적게 들고 사용자가 요구하고 있는 프라이버시 문제를 명쾌하게 해결할 수 있어 RFID 상용화에 이바지할 수 있을 것이라 생각된다.

1. 서 론

RFID(Radio Frequency Identification) 시스템은 물품에 RFID 태그를 부착하고, 이를 통한 물품의 인식 정보를 기반으로 주변의 모든 정보를 탐지하여 이를 실시간으로 네트워크에 연결하여 정보를 관리하는 것을 말한다. 이런 RFID 시스템을 적용하면 자동화 및 많은 이점을 얻을 수 있어 유통·물류 시장 등에서 큰 호응과 기대를 가져왔다. 하지만 사용자도 알지 못하는 사이에 모든 리더에게 자동적으로 응답하여 쉽게 식별된다는 RFID 태그의 특성 때문에 사용자 개인의 프라이버시문제(위치정보나 구매이력 노출 등)가 제기되면서 RFID의 상용화에 걸림돌이 되었으며, 업계와 학계에서는 이를 해결하는 것을 선결과제로 인식하고 있다.

프라이버시 문제를 해결하기 위해 Kill Command 기법, Hash-lock 기법, Re-Encryption기법 등 다양한 해결방안이 연구되었으나 이런 기법들을 현 RFID 시스템에 적용하는 것은 아직은 이른 감이 있다. 현 RFID 시스템을 변경해야하기 때문에 초기 도입 비용이 많이 든다는 문제점 이외에도 아직 해결해야할 다른 문제점들도 많기 때문이다. 본 논문에서는 현 RFID 시스템에 도입 가능하고 보안 모듈을 내장하기 힘든 저가의 태그들도 보호할 수 있는 보안 에이전트를 제안하고자 한다. 보안 에이전트에 대한 개념을 정립하고 등록된 태그만을 대상으로 개별적인 블로킹을 어떻게 수행하는지에 대해서도 살펴보도록 한다.

2. 관련연구

그동안 프라이버시문제를 해결하기 위하여 여러 가지 기법이 제안되었는데 이들을 자세히 살펴보도록 한다[1~3]. 제일 간단한 방법인 Kill Command 기법은 상품 구매 후, 태그에 Kill command를 보내 태그가 더 이상 리더에게 응답하지 않도록 하는 방법으로 현 상황에 쉽게 적용될 수는 있지만 태그를 재사용할 수 없다는 단점을 가지고 있다[1].

Hash-lock기법과 Randomized Hash-lock 기법은 해쉬 함수를 이용하는 방법으로, Hash-lock기법은 태그에 메타 ID가 저장되어 불특정 리더에게는 메타 ID를 응답하고 인증된 리더에게는 태그의 실제 ID를 응답하게 된다. 이 방법은 메타 ID를 사용하기 때문에 태그의 실제 ID를 쉽게 노출시지는 않지만 메타 ID가 변경되지 않아 사용자의 위치정보를 노출시킬 수 있다는 문제점을 안고 있다[1]. Randomized Hash-lock 기법에서는 위치정보보호를 위해 임의의 수 R을 이용하여 태그가 응답하는 메타 ID 값이 계속 변하도록 한다. 이 두 가지 기법은 태그에 해쉬 함수를 넣어야만 이룰 수 있는 방법으로 태그 제작비용의 증대라는 문제점을 안고 있다[1].

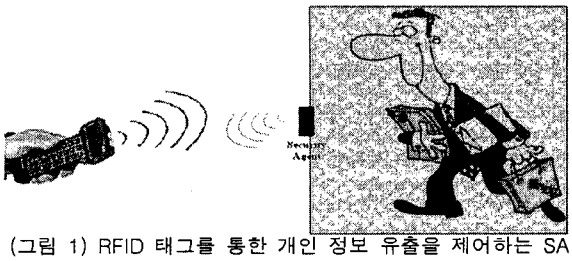
Re-encryption 기법은 태그에 암호화된 ID를 넣는 방법으로 주기적으로 적법한 리더가 암호 알고리즘을 사용하여 암호화된 ID를 변경한다[2]. 이 방법은 태그의 응답을 가지고 실제 ID를 쉽게 알 수 없고, 주기적으로 바뀌는 태그의 응답으로 위치정보보호를 이룰 수 있고 현재 보급된 태그에도 사용가능하다는 장점을 가지고 있지만, RFID 리더 및 시스템이 추가적인 기능을 지원해야만 한다는 문제점을 안고 있다.

마지막으로 Blocker Tag는 리더가 비트별로 요청 Q를 보낼 때 '0'과 '1'에 대해 모두 응답하여 리더가 주위에 많은 태그들이 존재한다고 생각하도록 하는 것으로, 실제로 리더는 주의에 있는 태그 ID 인식에 실패하게 된다[3]. 현재 적용이 가능하고 프라이버시 존(privacy zone)이라는 개념을 두어 선별적인 블로킹을 지원하려하고 있지만 우리가 원하는 기능인 개별적인 태그 블로킹은 지원하지 않는다.

3. 보안 에이전트

보안 에이전트란 컴퓨팅 능력이 적은 현재 RFID 시스템의 태그들을 대신하여 보안 메커니즘을 수행하는 보안 구성요소로서 기존의 리더와 태그로 구성되는 RFID 시스템에 별도로 추가되는 요소이다. 보안 에이전트의 개념을 그림으로 나타내면 (그림 1)과 같다.

보안 에이전트는 인증되지 않은 리더의 요청에 대한 태그의 응답을 차단하여 태그의 ID를 보호하며, 태그를 대신하여 리더와 인증 작업 등을 수행하고 적법한 리더라고 판단되면 태그의 응답을 차단하였던 것을 해제하여 태그가 자신의 ID를 리더에게 보내도록 한다. 이 때, 보안 에이전트는 등록과정을 거친



(그림 1) RFID 태그를 통한 개인 정보 유출을 제어하는 SA

복수 개의 태그에 대해 동작하며 처리 능력을 가지고 있고 소프트웨어 업그레이드가 가능한 기기로 구성하여 다양한 보안 기술을 처리할 수 있도록 한다.

본 논문에서 제안하고 있는 보안 에이전트는 보안 기능의 수준 및 부가 기능에 따라 다양한 시각으로 재해석될 수 있다. 첫 번째는 "Privacy Alarm"으로 동작할 수 있다는 것이다. 이는 보안 에이전트가 최소한의 동작만을 수행하였을 때의 보안 모델로서, 사용자에게 현재 자신이 소유한 태그의 정보를 읽고 있는 리더가 있음을 알려주어 프라이버시 문제에 대비할 수 있도록 한다. 필요에 따라 RFID 태그의 응답을 기록해두는 기능을 추가하게 되면 "Reader Monitor"로 발전할 수 있다.

둘째, "Tag Shielder"로 동작할 수 있다. 사용자는 개인정보가 유출될 수 있는 태그들을 사전에 "Tag Shielder"에 등록해 놓은 뒤, 집안이나 사무실 등의 안전한 지역을 벗어나게 될 때 "Tag Shielder"가 동작하도록 하여, 어떤 리더에게 요청이 왔을 때 등록된 태그의 응답을 차단한다. 셋째로 "Tag Shielder"에 RFID 리더와의 인증 프로토콜 등 보안 처리 능력을 추가하고, 차단 기능을 동적으로 스위칭 할 수 있도록 하면 앞에서 보았던 (그림 1)의 보안 에이전트가 된다.

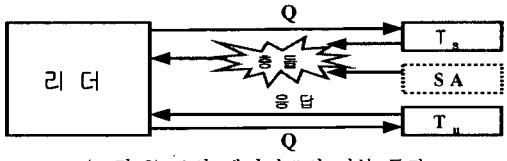
보안 에이전트는 태그를 대신하여 보안 모듈을 수행하여 태그를 보호하게 된다. 이와 같은 특징 때문에 보안 모듈을 내장하기 힘든 저등급의 태그도 보호할 수 있게 되고, 현재 보급되고 있는 태그뿐만 아니라 향후 다양한 태그들이 보급되었을 때에도 일관적으로 태그들을 보호할 수 있게 된다. 또 보안 에이전트를 통해 다양한 보안 모듈을 융통성 있게 적용해 볼 수 있고 새로운 보안 모듈이 제안되면 소프트웨어적으로 업그레이드도 가능하다. 또 보안 에이전트는 별도로 추가되는 구성요소이기 때문에 초기 도입 비용이 적게 든다는 경제적인 장점도 가지고 있다.

이렇게 프라이버시 보호를 위해 유용한 기능을 수행하는 보안 에이전트는 처리 능력을 겸비한 액티브 태그 형태로 동작하는 것을 목표로 하고 있지만, 기존에 개발된 휴대폰이나 PDA에 장착되는 리더를 이용해서 구현될 수도 있다. 최근 일본 휴대폰 업체인 KDDI에서 개발된 RFID 태그 리더 기능이 장착된 휴대폰을 이용할 수도 있고, 정보통신부에서 추진하고 있는 휴대폰에 RFID 리더 기능을 탑재하는 "모바일 RFID"를 이용하여 구현될 수 있을 것이다[4].

4. 선별적인 태그 블로킹

보안 에이전트는 등록된 태그들만을 대상으로 동작하기 때문에 선별적으로 태그를 블로킹하는 기법이 필요하다. T_s를 "보안 에이전트에 등록되어 보호대상이 되는 태그 집합"이라 하고, T_u를 "보안 에이전트에 등록되지 않은 보호가 필요 없는 태그 집합"이라고 정의하자. 그러면 T_s와 T_u는 ID 영역에 임의로 산재되어있는 태그의 집합으로서 "$T_s \cap T_u = \varnothing$"이다. 이 때, 보안 에이전트는 등록된 태그만을 선별적으로 블로킹하게 되는데 이를 그림으로 표현하면 (그림 2)와 같다. T는 보안 에이전트의 보호 범위에 있는 태그 집합($T=T_s \cup T_u$)이고 t_i는 T의 원소, 리더의 요청 Q에 대한 보안 에이전트의 동작을 $SA(Q)$라고 하면, 보안 에이전트의 기본 동작을 수식으로 표현한 것은 다음과 같다.

$$SA(Q) = \begin{cases} t_i\text{에 대한 충돌 응답}, & \text{if } t_i \in T_s \\ \text{무응답}, & otherwise \end{cases}$$



(그림 2) 보안 에이전트의 기본 동작

보안 에이전트가 등록된 각 태그의 개별 ID에 대해서만 충돌을 일으킨다면, 결국 인증되지 않은 리더가 충돌을 통해 보호 대상인 태그의 ID를 인식할 수 있게 된다. 이를 방지하기 위하여 ID 값과 일정 범위 안에 인접한 값들에 대해서도 충돌을 일으키면, 인증되지 않은 리더가 보호 대상인 태그의 개별 ID를 추측할 수 없게 된다. 이 때 사용된 보안 에이전트가 충돌 응답을 발생시키게 되는 ID 값의 범위 크기를 SR(Safe Range)의 개념이 필요하다. SR을 사용한 보안 에이전트의 동작을 식으로 표현하면 다음과 같다.

$$SA(Q, SR) = \begin{cases} t_z\text{에 대한 충돌 응답}, \\ \qquad \text{if } {}^{\forall}t_i \in T_s, f(S(t_z), S(t_i)) \le SR \\ \text{무응답}, otherwise \end{cases}$$

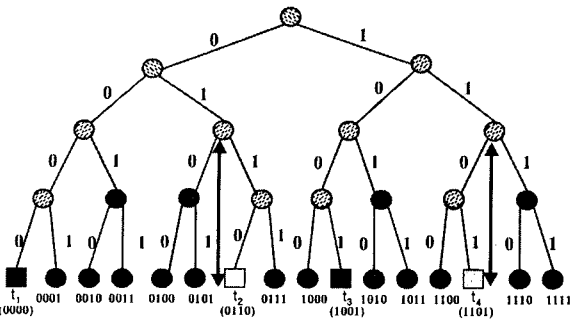
여기서 t_z는 보안 에이전트가 충돌 응답을 보내게 되는 일정 범위 안에 포함된 가상 태그이고, $S()$는 태그 Singulation 함수를 나타내며, $f()$는 편차를 나타내는 산술 함수이다.

5. 선별적인 태그 블로킹 알고리즘

Singulation 알고리즘은 리더가 요청 Q를 보냈을 때 여러 태그들이 동시에 응답하게 되면 응답 간에 충돌이 일어나서 리더가 태그의 응답을 제대로 인식할 수 없기 때문에, 태그의 정보를 하나씩 인식하기 위해 사용되는 알고리즘이다. 본 논문에서는 선별적으로 등록된 태그만을 블로킹하는 알고리즘을 대표적인 Singulation 알고리즘인 "Tree-Walking 알고리즘"에 적용하여 설명한다. 태그 블로킹을 처음 제안한 "Blocker Tag"에서 제안되었던 방법과 비교하여 설명하도록 한다.

일반적인 Tree-Walking 알고리즘은 리더가 비트별로 Q를 보내고 태그의 응답을 종합하여 주변의 태그를 인식하는 방법으로 이진 트리 탐색 방법과 유사하다[1,3]. (그림 3)을 예로 들어 설명해보자. 리더는 우선 '0'으로 시작하는 태그가 있는지 묻는 Q를 보낸다. Q에 해당하는 태그는 응답을 보내게 될 것이다. (그림 3)에서 볼 수 있듯이 "0000", "0110"이라는 ID를 갖고 있는 태그들은 자신이 '0'으로 시작한다고 응답을 보내게 된다. 그러면 리더는 응답한 태그 중에 다음 비트가 '0'인 태그가 있냐고 묻는 Q를 보내고 응답이 없으면 다음 비트가 '1'인 태그가 있냐고 묻는 Q를 보낸다. 이렇게 이진 트리 탐색을 하듯이 리더는 Q를 보내고, 탐색을 마친 뒤 리더가 태그의 응답을 종합하게 되면 "0000", "0110", "1001", "1101" 이런 ID를 가진 태그들을 인식하게 된다.

Tree-Walking에 대해 "Blocker Tag"가 사용하는 방법은 리더가 비트별로 Q를 보낼 때, Blocker Tag는 '0'과 '1'에 대해 모두 응답하여 리더가 주위에 있는 태그를 Singulation하는 것에 실패하여 태그의 ID를 인식하지 못하게 한다[3]. (그림 3)으로 설명해보자면, 실제 존재하고 있는 태그는 4개지만 리더



(그림 3) Tree-Walking에서의 SR

는 Singulation에 실패하여 "0000, 0001, ..., 1111" 이런 ID를 가지는 16개의 태그를 인식하게 되는 결과를 초래하게 된다. Blocker Tag는 이렇게 모든 Q에 대해 응답할 수도 있지만 프라이버시 존(Privacy zone, or prefix)이라는 개념을 두어 일정 공간을 블로킹할 수도 있다. 예를 들어, '1'로 시작하는 공간을 프라이버시 존으로 설정하면 루트를 기준으로 '1'로 시작하는 서브트리만을 블로킹하게 된다. 결과적으로 리더는 "0000, 0001, 0010, 0011, 0100, 0101, 0110, 0111" 이런 ID를 가진 8개의 태그가 존재한다고 생각하게 된다. 그러나 프라이버시 존에 보호 태그와 비보호 태그가 섞여있는 경우, 예를 들어 t_3은 보호대상이 아닌 태그이고 t_4는 보호 대상인 태그인 상황에서는 적용하기 어렵기 때문에 개별적인 태그 블로킹이 이루어진다고는 할 수 없다.

"Tree-Walking"에 대해 보안 에이전트가 사용하는 방법은 보안 에이전트에 등록된 태그의 ID를 정확하게 알 수 없도록 리더가 SR(Safe Range) 공간 안에 다른 태그들도 존재한다고 생각하도록 만드는 것이다. 이는 Tree-Walking 알고리즘이 동작하는 동안 Q가 SR에 도달하면 보안 에이전트가 무조건 응답하여 이룰 수 있다. 여기서 SR은 앞에서 살펴보았듯이 보안 에이전트가 충돌 응답을 발생시키는 ID 값 범위의 크기를 나타내는데, 이를 이진트리 측면에서 다시 정의해보면 보안 에이전트에 등록된 태그 ID를 포함하는 서브트리의 높이(depth)라고 할 수 있다. 또 태그 ID의 길이(비트) 측면에서는 보호하고자 하는 태그의 일정 ID 공간을 나타내는 하위 비트 길이라고 할 수 있다. (그림 3)에서 리프노드에 빗금으로 표시된 것이 등록된 태그를 나타내고 그림 상에서는 t_2와 t_4가 이에 해당한다. t_2와 t_4 옆에 있는 세로 화살표는 등록된 태그가 갖게 되는 SR이다.

Tree-Walking 알고리즘에서 보안 에이전트의 동작은 다음의 알고리즘으로 표현할 수 있다. T, T_s, t_i는 앞서 정의한 것과 같고, N은 태그 ID 길이(비트)이다. 리더는 요청 Q를 보내면서 지금까지 Singulation되었던 ID Prefix와 Q가 ID 공간 중 몇 번째 비트인지(j)를 알려주게 되고, 보안 에이전트는 T의 모든 원소 t_i에 대해 다음과 같은 연산을 수행한다.

t_i가 T_s의 원소이고 지금까지 Singualtion된 ID Prefix와 t_i의 ID Prefix가 같고 리더가 보낸 Q에 대해 응답하게 되는 j번째 비트가 SR 공간에 해당하면 보안 에이전트는 리더에게 '0'과 '1'을 모두 응답하여 충돌을 유발한다. t_i가 T_s의 원소이더라도 SR 공간에 해당되지 않거나 t_i가 T_s의 원소가 아니면 보안 에이전트는 아무런 동작도 수행하지 않는다.

보안 에이전트의 동작을 예를 들어 설명해보도록 한다. 앞의 (그림 3)에서와 같이 $T=\{t_1, t_2, t_3, t_4\}$, $T_s=\{t_2, t_4\}$이라 하고, 태그 각각의 ID는 t_1(0000), t_2(0110), t_3(1001), t_4(1101)이고, SR은 2라고 하자. 인증되지 않은 리더가 태그의 정보를 읽기 위

```
Selective_Block(Q)
/* Q.S_prefix ~ 현재까지 singulation된 ID prefix */
/* Q.j        ~ 현재 singulation하려는 bit order */
    for each t_i∈T_s
        /* 현재 singulation하려는 비트가 SR에 해당하면 */
        if (Q.j ≥ (N-SR))
                /* Q.S_prefix와 t_i의 prefix가 같으면 */
                if (prefix(Q.S_prefix,N-SR)==prefix(t_i,N-SR))
                        t_i에 대한 충돌응답 전송
                else
                        무응답
        else
                무응답
```

해 Tree-Walking 알고리즘을 수행하고 보안 에이전트가 동작하면, 실제로는 4개의 태그가 존재하지만 리더는 "0000, (0100, 0101, 0110, 0111), 1001, (1100, 1101, 1110, 1111)" 이런 ID를 가진 10개의 태그를 인식했다고 생각하게 된다. 이렇게 보안 에이전트는 등록된 태그의 ID에 인접한 일정범위에서 충돌을 유발하고 리더가 태그의 ID를 정확하게 알 수 없도록 방해하여 등록된 태그를 Singulation하는데 실패하도록 한다.

6. 기대효과 및 결론

본 논문에서 제안하고 있는 보안 에이전트는 태그를 대신하여 보안 모듈을 수행하는 추가 구성요소이다. 추가 구성요소라는 특징 때문에 현 RFID 시스템의 변경 없이 바로 도입 가능하고, 저등급의 태그들도 보호할 수 있다. 이렇게 보안 에이전트 개념을 도입하면 기존에 가지고 있던 RFID 보안 체계의 변경을 유발할 수 있고, 저비용으로 효율적인 태그 보호를 수행할 수 있어 한층 빨리 RFID 시스템이 상용화될 수 있을 것으로 보인다. 또 개인정보보호에 대한 인식도 좋아질 것으로 기대된다.

본 논문에서는 Singulation 알고리즘 중 Tree-Walking에 대해서만 설명했지만 slotted ALOHA에서도 비슷한 방법으로 적용 가능하다. 결과적으로 보안 에이전트는 보호 대상인 태그만을 개별적으로 블로킹할 수 있게 된다.

보호대상이 아닌 태그가 SR의 범위 안에 포함되어 블로킹의 대상이 되었을 때에는 SR을 동적으로 조정하는 것으로 이를 해결할 수 있을 것이다. 동적으로 SR을 조정하는 기법에 대해서는 향후 연구과제로 남겨놓는다.

참고문헌

[1] Stephen A. Weis, "Security and Privacy Aspects of Low-Cost Radio Frequency Identification Systems", Security in Pervasive Computing 2003, LNCS 2802, pp.201-212, 2004

[2] Junichiro Saito, "Enhancing privacy of Universal Re-encryption scheme for RFID tags", Embedded and Ubiquitous Computing: International Conference EUC 2004, 2004

[3] Ari Juels, "The Blocker Tag: Selective Blocking of RFID Tags for Consumer Privacy", Proceedings of the 10th ACM conference on Computer and communications security, pp.103-111, 2003

[4] KDDI, http://www.kddi.com/english/corporate/news_release/2005/0302/index.htm