논문 2010-47TC-1-13

충돌방지 알고리즘의 보안 견고성

(Security Robustness of Tree based Anti-collision Algorithms)

서 현 곤*, 김 향 미**

(Hyun-gon Seo and Hyang-Mi Kim)

요 약

RFID(Radio Frequency IDentification) 기술은 RF신호를 사용하여 물품에 부착된 전자태그를 식별하는 비접촉 기술이다. RFID 리더의 식별 영역 내에 여러 개의 태그가 있는 경우 이들 사이의 충돌(collision)이 발생되기 때문에 이들을 식별할 수 있는 메커니즘이 필요하다. 다중 태그 식별 문제는 RFID 기술 중에서도 핵심이며, 이 문제는 충돌방지(anti-collision) 알고리즘을 통하여 해결할 수 있다. 하지만 RFID 시스템의 또 다른 문제는 정보보호이다. 태그는 리더의 쿼리에 매우 쉽게 응답하기 때문에 태그의 정보 노출에 따른 사용자 프라이버시 침해 문제가 발생한다. 이러한 점에서 RFID 기술은 외부로부터 스니핑(sniffing)에 매우 취약하다. 본 논문에서는 기존에 제안된 트리 기반 메모리래스 알고리즘인 트리-워킹 알고리즘, 쿼리 트리 알고리즘, 향상된 쿼리 트리 알고리즘 등의 보안 견고성에 대하여 살펴본다.

Abstract

RFID(Radio Frequency IDentification) is a technology that automatically identifies objects containing the electronic tags by using radio wave. When there are some tags in the domain of the RFID reader, the mechanism that can solve a collision between the tags occurs is necessary. The multi tag identification problem is the core issue in the RFID and could be resolved by the anti-collision algorithm. However, RFID system has another problem. The problem id user information security. Tag response easily by query of reader, so the system happened user privacy violent problem by tag information exposure. In the case, RFID system id weak from sniffing by outside. In this paper, We study of security robustness for tree-walking algorithm, query tree algorithm and advanced query tree algorithm of tree based memoryless algorithm.

Keywords: RFID, Anti-Collision algorithm, Security, Robustness, Multi Tag

Ⅰ. 서 론

RFID(Radio Frequency Identification)는 유비쿼터스 컴퓨팅 환경의 초기 기반 기술로, 물체에 전자태그를 부착하고 RF 신호를 이용하여 물체의 태그 정보를 읽어와 물체를 식별하는 비접촉 인식기술이다. 무선주파수 인식 기술의 중요성이 점차 커지고 유비쿼터스 컴퓨팅 기반 기술이 대두되면서 비접촉식 대용량 데이터의 전송이 가능한 RFID 기술은 이미 다양한 분야에 적용되고 있다. 정보의 실시간 처리와 네트워크화의 특징으로 유통 및 물류 관리, 보안, 안전, 소방·방재, 환경 관리는 물론, 생산자동화, 금융서비스, 우정 등의 분야에서 RFID가 바코드를 대체할 기술로 주목받고 있는 것은 당연한 결과이다.

그러나 RFID 기술의 확산을 위해 해결해야 할 몇 가지 문제점들로 태그의 저가격, 저전력, 초소형화 문제와 다중 태그들의 충돌 문제, 그리고 사용자 프라이버시 및 정보보호 문제 등이 있다. 기술의 발달로 인해 태그의 저가격, 저전력, 초소형화는 점차 해결되고 있으며 다중 태그들의 충돌 문제 역시 그 해결방안으로 여러 가지 충돌방지 알고리즘이 제안되고 있다. 하지만 앞선 문제들이 기술적으로 해결되며 태그의 크기가 작아지면 질수록 인식거리가 멀어지면 질수록 소비자는 RFID의 부착여부를 식별하기가 극히 어려워진다. 더군다나

* 정회원, ** 학생회원, 한라대학교 정보통신방송공학부
(Halla University)

※ 이 연구는 2008년도 한라대학교 교내연구비 지원에 의한 것임

접수일자: 2009년8월11일, 수정완료일: 2010년1월18일

RFID 태그는 생산된 개별 상품에 대하여 식별 코드가 유일하게 부여되는 특징이 있기 때문에 소비자 개별적인 추적이 가능해지고 개인 정보의 유출로 이어질 수 있다. 이러한 개인정보의 유출은 현 시점에서 RFID 기술의 확산을 저해하는 가장 큰 요인이라고 말할 수 있다[1~3].

본 논문에서는 RFID 시스템의 개인 프라이버시 및 정보보호를 위한 연구의 일환으로 기존에 제안되어 있는 트리기반 알고리즘의 보안 견고성에 대하여 연구한다. 본 논문의 구성은 다음과 같다. Ⅱ장에서는 관련연구로써 태그 정보 보호 기술들과 트리기반 충돌방지 알고리즘들에 대해 살펴보고, Ⅲ장에서 트리기반 충돌방지 알고리즘들의 보안 견고성에 대하여 연구하고 결과를 비교한다. 그리고 Ⅳ장에서 결론을 맺는다.

Ⅱ. 관련 연구

본 장에서는 현재까지 연구된 태그의 정보보호기술들과 트리기반 충돌방지 알고리즘에 대해 살펴본다.

1. 태그 정보 보호 기술

Kill Command 기술은 RFID 프라이버시 보호를 위한 가장 극단적인 방법으로 사용자가 물건을 구매하고 나면 출구에서 RFID 태그를 파괴하거나 'Kill명령어'를 작동시켜 태그의 사용을 영구적으로 중지시키는 가장 간단하고 확실한 방법이다. 하지만 태그의 정보를 유지해야 하는 경우에 비춰볼 때는 매우 취약한 기술이다. 물건에 금속성의 박막을 입혀 무선주파수가 침투하지 못하도록 하는 Faraday Cage 방법은 Kill 명령어와 달리 박막을 벗기면 재사용이 가능하나 상품의 절도에 악용될 수 있다.

액티브 jamming은 태그 통신을 차폐시키는 물리적인 방법인 Active Jamming Approach 기술로 고객이 부근에 있는 RFID 리더의 동작을 방해하거나 차단시키기 위해 능동적으로 무선 신호를 방송하는 장치를 가지고 다니며 방해전파를 보내야한다. 하지만 주위의 모든 RFID 시스템을 교란시키고, 프라이버시에 관심이 없는 합법적인 시스템도 방해할 수 있다.

RFID 태그와 리더의 교신은 간단히 도청되므로 통신에 암호화를 적용하거나 태그의 소유자나 특정 리더만 통신이 가능하도록 인증을 이용하는 인증기법이 있다. 리더의 능력은 제한적이지 않기 때문에 태그에 써넣을 내용을 암호화하여 써넣는 것이 가능하지만 저가형으로 한정된 계산능력 밖에 탑재할 수 없는 RFID 태그가 이러한 복잡한 암호화를 처리한다는 것은 현실적으로 어렵다. 암호화된 RFID 태그 정보를 다시 정기적으로 암호화하는 재암호화 기법을 이용하면, RFID 태그의 정보는 재암호화에 의해 정기적으로 고쳐 써넣으므로, 특정의 RFID 태그 정보를 추적하는 것을 방지할 수 있고, RFID 태그의 복잡한 처리 없이 RFID 태그의 비용을 낮추는 것이 가능한 이점이 있다. 그러나 특정 공개키에 관련된 RFID 태그를 추적하는 것이 가능하기 때문에 위치정보가 침해되어 버린다.

태그-리드간의 통신모델에서 통신 계층에 사용되는 트리워킹 방식의 충돌 방지 프로토콜을 역이용하는 Blocker Tag 기법은 트리워킹 알고리즘 수행 중 '보호구역'으로 지정된 프리픽스(prefix)를 가지는 상태에 들어가게 되면 "블로커 태그"는 자신의 실제 ID와 상관없이 충돌을 발생시킨다. RFID 전체 길이 중 프리픽스를 제외한 부분이 충분히 크다면 리드는 실제로는 트리워킹 알고리즘을 수행할 수 없을 정도로 많은 단계에서 충돌을 겪게 된다. 이렇게 함으로써 인증되지 않은 리드가 태그에 접근하는 것을 방지하게 된다. 그러나 사용자가 "블로커 태그"라는 도구를 가지고 다녀야 하는 단점이 있다.

Hash Lock 방법을 사용하기 위해서는 태그에 해시함수가 구현되어 있어야하며, 임시적으로 metaID를 저장할 수 있는 메모리 공간이 제공되어야 한다. 공격자가 태그에게 질의를 해서 태그의 metaID를 받은 다음 적법한 리더에게 스프핑한 태그의 metaID를 전달하게 되면, 리더가 스프핑된 태그에게 올바른 key를 노출하는 위험이 있다. 또한 metaID가 여전히 식별자의 역할을 수행하기 때문에 사용자가 추적될 수 있는 문제점이 있다. Hash Lock 기법의 단점을 보완하여 제안한 Randomized Hash Lock 기법은 태그가 추적되는 문제를 방지하기 위해 추가 연산을 적용한 것이다. 이 기법은 적은 수의 태그를 사용하는 사용자에게는 적합한 반면 리더가 잠근 태그가 많을수록 태그의 잠김 상태를 해제하는 데 많은 시간이 걸리기 때문에 많은 수의 태그를 가진 경우에 사용하기에는 부적합 하다. Hash Based Authentication 기법은 해시를 기반으로 하는 인증 기법으로 프로토콜 수행과정 중에 태그가 두 번의 해시연산을 하게 되어 있다. 호스트에 계산 부하가 매우 적으나 보안상의 많은 허점을 가지고 있다. Hash Chain Based 기법은 태그 내부에 두 개의 일방향(one-way) 해시함수를 가지고 있다고 가정하고, 리드의

질의에 대해 태그는 프로토콜을 수행할 때마다 두 번의 해시 계산을 함으로써, 외부에서 예측 불가능한 값을 송신하고, 태그 내부의 비밀 값을 스스로 갱신한다. 외부로부터의 입력 값이 없기 때문에 프로토콜의 허점을 이용한 공격이 어렵고 태그 스스로가 자신의 내부 정보를 갱신하는 방식을 취하기 때문에 도청될 위험이 없지만 호스트에서 태그를 식별하기 위한 계산양이 너무 많다. Hash관련 보안 기법들의 큰 단점은 가격이다.

현재까지 프라이버시 조건을 완벽하게 만족시킬 수 있는 한 가지 방법은 없다. 여러 가지 방법을 조합하여 사용하는 것이 가장 좋은 방법이라 할 수 있겠다[4].

2. 트리 기반 충돌 방지 알고리즘

리더기와 다수의 태그 간에 하나의 동일 채널을 이용하기 때문에 발생하는 충돌의 문제와 동일 식별 영역 내에 다수의 태그가 존재할 경우에 요구되는 다중 태그 식별 문제는 RFID 기술 중에서도 핵심이며 이 문제는 충돌방지(anti-collision) 알고리즘을 통하여 해결할 수 있다[13]. 현재까지 다양한 충돌방지 알고리즘들이 제안되었는데 크게 트리 기반 알고리즘(tree based algorithms)과 슬롯 알로하 기반 알고리즘(slot aloha based algorithms)으로 구분된다.

슬롯 알로하 기반 알고리즘은 확률적(probabilistic) 알고리즘으로 슬롯간의 시간차를 이용하여 태그 충돌을 회피한다. 하지만 적절한 슬롯의 개수와 종료시점에 따라 태그 인식률이 종속되기 때문에 정확하게 모든 태그를 파악한다고 말하기 어렵다.

이에 비해 트리 기반 알고리즘은 결정적(deterministic) 알고리즘으로 이진 트리를 순회하며 모든 태그를 식별한다. 트리 기반 알고리즘은 메모리형(memory) 알고리즘과 메모리래스형(memoryless) 알고리즘으로 구분하는데, 태그에 대한 질의만으로 태그를 구분할 경우 이는 메모리래스형 알고리즘이 되고 리더기의 질의에 대해 태그마다 상태 정보를 저장하고 관리에 의하여 태그를 식별할 경우 이를 메모리형 알고리즘이라 한다. 메모리래스형 알고리즘은 리더기가 태그들의 식별을 위해 반복적인 질의, 응답과정을 통하여 이루어지기 때문에 태그에는 인식을 위한 별도의 기억 장치가 필요 없어 저가격, 저전력, 초소형화를 실현할 수 있다[5].

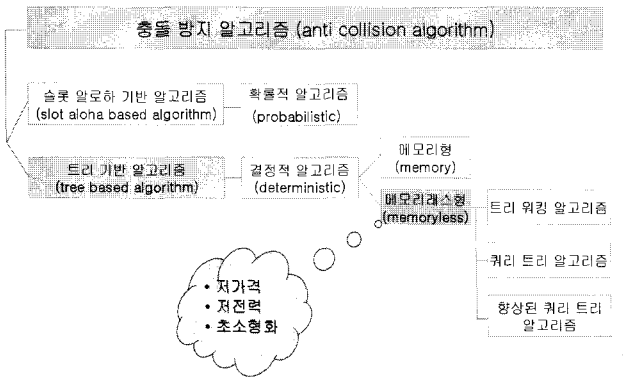

그림 1. 충돌 방지 알고리즘
Fig. 1. Anti collision Algorithm.

본 장에서는 충돌방지를 위한 트리기반 알고리즘인 트리 워킹 알고리즘, 쿼리 트리 알고리즘, 향상된 쿼리 트리 알고리즘의 특징과 동작원리에 대하여 살펴본다.

가. 트리 워킹(TW : Tree-walking) 알고리즘

트리 워킹 알고리즘은 대표적인 메모리래스 알고리즘이다. bit-by-bit 쿼리 방식이며 이진트리의 깊이 우선 탐색방법으로 태그를 식별한다[1, 5].

리더기가 d비트 길이의 프리픽스$\mathcal{B}(\mathcal{B}=b_1b_2, \ldots, b_d)$를 모든 태그에게 질의하면, 태그들은 자신의 식별자와 프리픽스를 비교하여 그 값이 동일할 경우 태그는 $d+1$ 비트를 리더기에 전송한다. 모든 태그가 "0" 또는 "1"을 전송했을 경우, 충돌이 발생한 것이 아니기 때문에 리더기는 새로운 프리픽스 $\mathcal{B}_{d+i}$를 생성하여 다시 모든 태그에게 전송한다. 하지만 "0"과 "1"이 동시에 리더기에 수신될 경우 충돌이 발생하기 때문에 리더기는 프리픽스($\mathcal{B}$)에 "0"과 "1"을 추가해서 새로운 프리픽스"$\mathcal{B}0$", "$\mathcal{B}1$"를 생성하여 다시 질의를 반복한다. 트리 워킹 알고리즘의 경우 비트단위로 태그를 검출하기 때문에 리더기 영역 내에 태그가 하나만 있는 경우도 마지막 비트까지 질의($\mathcal{Q}$)-응답($\mathcal{R}$)을 해야 하고, 영역 내에 많은 태그가 있을 경우 충돌이 자주 발생하므로 질의 시간에 의한 성능저하의 문제점을 가지고 있다.

그림 2는 트리 워킹 알고리즘의 동작원리를 보여준

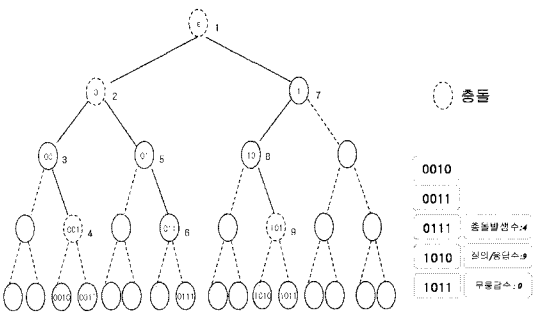

그림 2. Tree Walking Algorithm의 동작원리
Fig. 2. Principle of tree walking algorithm.

다. 첫 번째 과정에서 리더기가 모든 태그를 대상으로 질의(ε)하면 모든 태그들이 자신의 첫 번째 비트로 응답을 하게 되어 충돌이 발생한다. 두 번째 질의(Q)-응답(R) 과정에서 리더기가 프리픽스 '0'을 인자로 태그에 질의하면 영역내의 '0010', '0011', '0111'의 식별자를 가진 태그가 매치되면서 해당 식별자의 프리픽스 다음 비트인 '0', '0', '1'을 리더기로 전송한다. 이때 '0'과 '1'이 동시에 수신되므로 충돌이 발생한다. 충돌이 발생하면 리더기는 프리픽스('0')를 스택에 저장하고 동시에 '0'을 추가하여 새로운 프리픽스('00')를 생성한 후 다시 질의(Q)-응답(R) 과정(세 번째)을 시작한다. 이때 영역내의 '0010', '0011'의 식별자를 가진 태그가 매치되면서 각 식별자의 프리픽스 다음 비트인 '1'을 리더기로 전송하다. 이때 리더기는 모두 '1'을 수신하므로 충돌이 발생하지 않고 프리픽스에 '1'을 추가하여 새로운 프리픽스('001')를 생성한다. 네 번째 질의(Q)-응답(R) 과정에서 프리픽스 '001'에 매치되는 식별자 '0010', '0011'이 리더기로 응답한다. 이때 '0'과 '1'이 동시에 수신되면서 충돌이 발생하나 프리픽스의 길이가 태그 식별자 비트 길이 보다 하나 작은 3이므로 리더기는 단순히 영역 내에 '0010'과 '0011'의 식별자를 가진 두 개의 태그가 존재한다고 판단하게 된다. 따라서 이때 기존 프리픽스를 스택에 저장할 필요가 없으며 다만 새로운 프리픽스를 생성하기 위하여 스택에 저장된 프리픽스를 가져온다. 이때의 프리픽스는 두 번째 과정에서 충돌이 발생하여 스택에 저장된 '0'이며 여기에 '1'을 추가하여 새로운 프리픽스 '01'을 생성하고 5번째 질의(Q)-응답(R) 과정을 시작한다. 이와 같이 영역 내의 모든 태그를 식별하기 위하여 아홉 번의 질의(Q)-응답(R) 과정이 수행된다.

트리 워킹 알고리즘의 장점은 모든 태그를 빠짐없이 찾아낼 수 있다는 것이다. 그러나 시간이 너무 오래 걸린다는 취약점을 가지고 있다.

나. 쿼리 트리(QT : Query Tree) 알고리즘

쿼리 트리 알고리즘은 트리 워킹보다 향상된 방법이다. 리더기가 d비트 길이의 프리픽스 $\mathcal{B}(\mathcal{B}=b_1b_2, \ldots, b_d)$를 모든 태그에게 질의하면, 태그들은 자신의 식별자와 프리픽스를 비교하여 그 값이 동일할 경우 태그는 $d+1$비트에서 마지막 n비트까지 리더기에 전송한다[2, 5]. 리더기가 한 태그에게만 응답을 받을 경우 $\mathcal{B}_{d+1\ldots n}$태그가 식별된 것이고, 다수의 태그가 동시에 응답했을 경우 충돌이 발생한 것이므로 리더기는 프리픽스($\mathcal{B}$)에 "0"과 "1"을 추가해서 새로운 프리픽스"$\mathcal{B}0$", "$\mathcal{B}1$"를 생성하여 다시 질의를 반복한다. 하지만 쿼리 트리의 경우 리더기의 질의에 대해 무응답인 경우가 발생하고 충돌이 많이 발생하기 때문에 이를 처리하기 위한 부담이 증가한다.

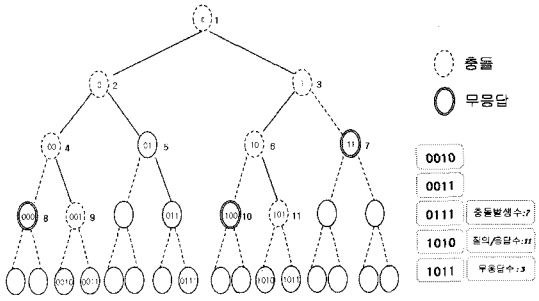

그림 3. Query Tree Algorithm의 동작원리
Fig. 3. Principle of query tree algorithm.

그림 3은 쿼리 트리 알고리즘의 동작원리를 보여준다. 질의(Q)-응답(R) 과정 중 네 번째 단계에서 리더기가 프리픽스 '00' 값을 인자로 태그에 질의하면 영역 내의 '0010', '0011'의 식별자를 가진 두 개의 태그가 매치되면서 각 식별자의 프리픽스 다음 비트부터 식별자의 마지막 비트까지인 '10', '11'을 리더기로 전송한다. 이때 리더기는 충돌을 인지하며 기존 프리픽스('00')에 '0'과 '1'을 찾아내어 각각 큐에서 새로운 프리픽스 "01"을 인자로 새로운 질의(Q)-응답(R) 과정(다섯 번째)을 시작하게 된다. 이때, 영역내의 '0111'의 식별자를 가진 태그가 매치되면서 식별자의 프리픽스 다음 비트부터 식별자의 마지막 비트까지의 '11'을 리더기로 전송한다. 이때 충돌은 발생하지 않으며 리더기는 '0111'의 식별자를 가진 태그를 식별하게 된다. 영역 내의 나머지 태그들을 식별하기 위해 리더기는 큐에 저장된 프리픽스를 가져와 새로운 질의(Q)-응답(R) 과정을 반복한다.

쿼리 트리 알고리즘의 경우 리더기의 질의에 대해 무응답인 경우가 발생하고 충돌이 많이 발생하여 이를 처리하기 위한 부담이 증가하게 된다. 또한 Attacker가 충돌과 무응답의 차이를 알 수 없다.

다. 향상된 쿼리 트리 기반(AQT : Advanced Query Tree) 알고리즘

향상된 쿼리 트리 기반 알고리즘은 트리 워킹 알고리즘, 쿼리 트리 알고리즘과 같은 충돌 회피 알고리즘이면서, 충돌이 발생한 비트의 위치를 리더기가 검출하는 충돌 검출 기능을 함께 가지고 있다[5]. 리더기가 d비트 길이의 프리픽스 $\mathcal{B}(\mathcal{B}=b_1b_2, \ldots, b_d)$를 모든 태그에게 질의하면, 태그들은 자신의 식별자와 프리픽스를 비교

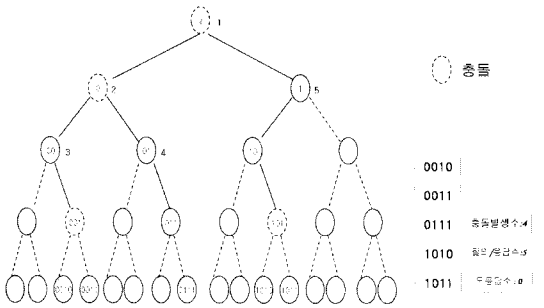

그림 4. Advanced Query Tree Algorithm의 동작원리
Fig. 4. Principle of advanced query tree algorithm.

하여 그 값이 동일할 경우 태그는 $d+1$비트에서 마지막 n비트까지 리더기에게 전송한다. 리더기가 하나의 태그에게만 응답을 받을 경우 쿼리 트리와 같이 하나의 태그가 식별된다. 하지만 다수의 태그가 동시에 응답하여 충돌이 발생하게 되었을 때, 응답한 $Bd+1$, ... n사이에서 최초 충돌이 발생한 비트의 위치인 k비트를 찾고 새로운 프리픽스(B)에 "$Bk0$"과 "$Bk1$"을 생성한다.

향상된 쿼리 트리 알고리즘의 기본 알고리즘은 쿼리 트리 알고리즘과 동일하며 깊이 우선의 탐색을 한다. 그림 4의 첫 번째 질의(Q)-응답(R) 과정에서 리더기가 질의하면 영역 내의 모든 태그들이 응답하여 충돌이 발생한다.

두 번째 질의(Q)-응답(R) 과정에서 리더기가 프리픽스 '0'을 인자로 태그에 질의하면 영역 내의 '0010', '0011', '0111'의 식별자를 가진 3 개의 태그가 매칭이 되면서 각자 식별자의 프리픽스 다음 나머지 비트 '010', '011', '111'을 리더기로 전송한다. 이 과정에서 Q와 R은 각각 1이 되며 '0'과 '1'이 동시에 수신되므로 충돌이 발생하게 된다. 충돌이 발생하면 리더기는 충돌 발생 위치를 찾아내어 프리픽스('0')를 저장하고 동시에 '0'을 추가하여 새로운 프리픽스('00')를 생성한 후 다시 질의(Q)-응답(R) 과정(세 번째)을 시작한다. 이 과정에서 리더기가 프리픽스 '00'을 인자로 태그에 질의하면 영역 내의 '0010', '0011'의 식별자를 가진 태그가 식별자 비트 길이 보다 하나 작은 3으로써 리더기는 영역 내에 두 개의 태그가 존재한다고 판단하게 된다.

III. 트리 기반 충돌 방지 알고리즘의 보안 견고성

본 장에서는 기존에 제안된 트리기반 충돌 방지 알고리즘의 보안 견고성에 대하여 알아본다. 모든 상황은 다음 가정 하에 연구되었다.

가정 : **공격자(Attacker)는 태그의 정보 비트수(k)를 알고 있다.**

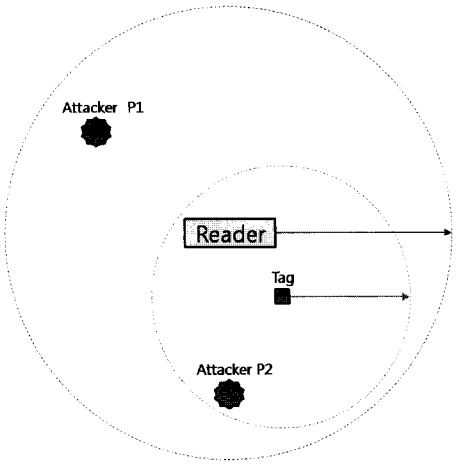

그림 5. RFID 시스템 상에서 Attacker의 위치
Fig. 5. Location of attacker in RFID system.

그림 5에서 'Attacker P1'은 리더의 송신 범위에 포함되어있지만 태그의 송신 범위로부터 벗어나 있다. 즉, 리더로부터의 쿼리는 인식할 수 있지만 태그로부터의 응답은 인식할 수 없는 상황이다. 하지만 'Attacker p2'는 리더의 송신 범위와 태그의 송신 범위에 모두 포함되는 지점에 위치해 있다. 이 경우, 리더의 쿼리와 태그로부터의 응답을 모두 인식할 수 있는 상황이다. 각 알고리즘의 보안 견고성을 판단함에 있어 Attacker의 위치가 태그의 송신 범위에 포함되는지 여부는 매우 중요한 요소로 작용한다.

본 논문에서는 'Attacker P1'의 상황과 'Attacker P2'의 상황을 앞서 말한 가정과 더불어 각 알고리즘의 보안 견고성을 판단하는 기준으로 제시한다.

가. 트리 워킹(Tree Walking) 알고리즘

수식 표현을 위해 다음의 기호를 사용한다. n은 존재하는 전체 태그의 수를 나타내며 k는 태그의 비트 수(크기), m은 받은 프리픽스 중에서 비트수가 $k-1$인 프리픽스의 개수, e는 추가로 선정된 태그의 수(overhead)를 나타낸다. 함수식 $f(k)$는 Attacker가 각 알고리즘을 통해 찾은 태그(finding tag)의 전체 개수를 의미한다.

(1) Attacker가 tag의 범위 밖에 존재할 경우

이 경우 Attacker는 리더에서 전달되는 모든 프리픽스는 들을 수 있지만, 태그가 응답하는 비트는 들을 수 없으므로 리더의 프리픽스만으로 영역 내에 있는 태그

를 추론하여야 한다.

이진트리의 깊이 우선 탐색방법으로 태그를 식별하는 트리 워킹 알고리즘에서는 마지막 비트($k-1$)에서 충돌이 발생하였을 경우 0과 1이 동시에 응답하였다는 것이므로 이때 무조건 2개의 태그를 모두 찾을 수 있다. 그러나 마지막 비트에서 충돌이 발생하지 않았을 경우 역시 2개의 태그를 찾는데 그 이유는 Attacker는 마지막 비트($k-1$)에서는 충돌여부를 알 수 없기 때문이다. 그림 6의 ST_k는 마지막 비트($k-1$)에서 형성된 서브트리로 Tag_{n-1}, Tag_n 두 개의 태그를 찾는다는 것을 그림을 통해 확인해볼 수 있다. 결국 Attacker는 무조건 $m\times2$만큼의 태그를 찾으므로 추가로 선정된 태그의 수 e(overhead)는 $m\times2$에서 실제 존재하는 태그의 수 n을 뺀 만큼임을 알 수 있다. 그러므로 찾는 태그의 수 ($f(k)$)는 존재하는 전체 태그의 수에 추가로 찾아낸 태그의 수를 더한 값($n+e$)이 된다.

여기서 가장 중요한 것은 Attacker가 마지막 프리픽스에서 충돌의 여부를 알 수 없다는 것이다. 하지만 리더의 프리픽스만으로도 영역 내에 존재하는 모든 태그 정보를 알 수 있기 때문에 보안에 취약함을 알 수 있다. 함수식은 수식(1)과 (2)와 같이 표현 가능하다.

$$f(k)=m\times2 \tag{1}$$

$$e=(m\times2)-n \tag{2}$$

Ⅱ장 2절의 그림 2의 사례를 수식에 적용해보면 4비트 크기의 다중태그 5개이므로 n은 5, k는 4이며 태그로부터 수신된 프리픽스 중에서 비트수가 k보다 하나 작은 즉 3인 개수 m은 3이다. 그러므로 Attacker가 찾은 태그의 수 $f(k)$는 6이고, 오버헤드인 e의 값은 1임을 알 수 있다.

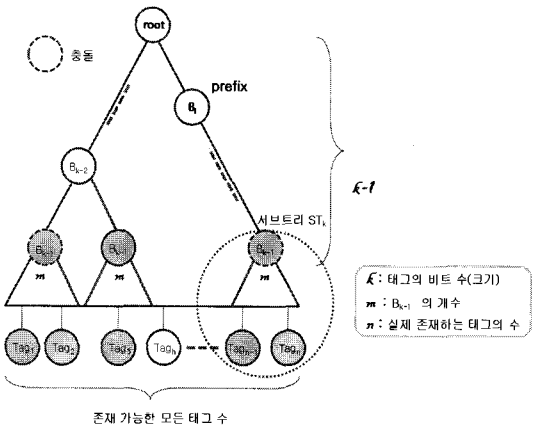

그림 6. 트리 워킹 알고리즘 이진트리의 형태(범위 밖)
Fig. 6. Tree walking algorithm binary tree(outside).

(2) Attacker가 tag의 범위 내에 존재할 경우

이 경우 리더의 프리픽스와 태그의 응답을 이용하여 태그의 값을 정확히 찾아 낼 수 있다. 마지막 프리픽스, 즉 $k-1$에서의 충돌 여부를 Attacker가 모두 알고 있다는 것이다. 그림 7에서 볼 수 있듯이 c_0는 ($k-1$)bit에서 충돌이 없는 프리픽스의 개수를 c_1는 ($k-1$)bit에서 충돌이 있는 프리픽스의 개수를 의미하며 Attacker가 찾는 태그의 수 $f(k)$를 함수식으로 표현하면 수식(3)과 같은 식이 가능하다.

$$f(k)=\{c_0+(c_1\times2)\}+e \tag{3}$$

$e=0$

$$f(k)=c_0+(c_1\times2) \tag{4}$$

$$f(k)=n \tag{5}$$

Ⅱ장 2절의 그림 2의 사례를 적용해보면, 충돌이 없는 프리픽스의 개수 즉 c_0는 1이고, c_1 즉 충돌이 있는 프리픽스의 수는 2이다. 그러므로 Attacker가 찾은 태그의 수 $f(k)$는 $1+2\times2=5$로 태그의 실제 존재하는 수 다섯 개와 일치하며 오버헤드 값은 0임을 알 수 있다. 즉 Attacker는 합법적인 RFID 리더가 알 수 있는 태그의 개수와 동일한 정보를 유추할 수 있어 보안에 매우 취약함을 알 수 있다.

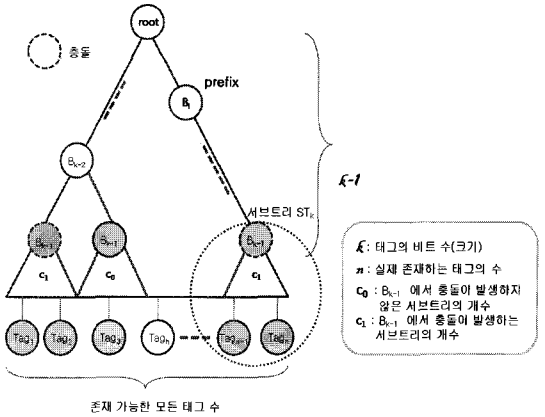

그림 7. 트리 워킹 알고리즘 이진트리의 형태(범위 내)
Fig. 7. Tree walking algorithm binary tree(inside).

나. Query Tree 알고리즘

쿼리 트리 알고리즘은 트리 워킹보다 향상된 방법이다. 태그가 리더에게 정보를 줄때 Tree Walking이 한 비트 씩 전달했다면 Query Tree는 리더가 질의를 할

때 태그가 가지고 있는 나머지 비트 정보를 모두 리더에게 보내는 것이다.

수식표현을 위해 다음의 기호를 사용한다. n은 존재하는 전체 태그의 수를 나타내며 k는 태그의 비트 수(크기), m은 받은 프리픽스 중에서 비트수가 $k-1$인 프리픽스의 개수, e는 추가로 선정된 tag의 수(overhead)로 표현한다. i는 하나의 태그만 존재하는 서브 트리의 수를 각각 나타낸다.

(1) Attacker가 tag의 범위 밖에 존재할 경우

이 경우는 Attacker의 입장에서 태그의 범위 밖에 존재하므로 충돌인지 무응답인지 판별할 수 없다. 또한 e는 항상 n보다 크다. 프리픽스 $k-1$에서는 무응답이든 충돌이든 항상 ×2만큼의 태그를 찾으므로 충돌로 둘 다 존재한다면 실재 태그의 수가 같을 수 있겠지만, 무응답일 경우엔 ×2만큼의 e(overhead)가 생기는 것이다.

또한 만약 $k-1$이전에 무응답이 발생하였다면 e의 개수는 더 클 수밖에 없다. 이때는 $k-1$비트를 제외한 프리픽스 중 자신이 받은 프리픽스의 비트수와 다음 받은 프리픽스의 비트수를 비교하여 후자가 작은 경우 현재 프리픽스에 속하는 태그는 한 개로 본다. 함수식은 다음과 같이 표현 가능하다.

$$f(k)=2^k \tag{6}$$

$$e=2^k-n \tag{7}$$

Ⅱ장 2절의 그림 3에서 $f(k)$는 $2k=24=16$이며, overhead 값(e)는 11이다. $2k$만큼의 태그를 찾아내므로 존재할 수 있는 모든 태그를 찾는다. 그림 8에서도 확인할 수 있듯이 리더는 Tag_1~Tag_n까지 존재가능한 모든 태그를 찾게 된다. 다시 말해 이것은 실재 존재하는 태그가 정확히 무엇인지 알 수 없다는 이야기로 보안 견고성이 뛰어나다고 할 수 있다.

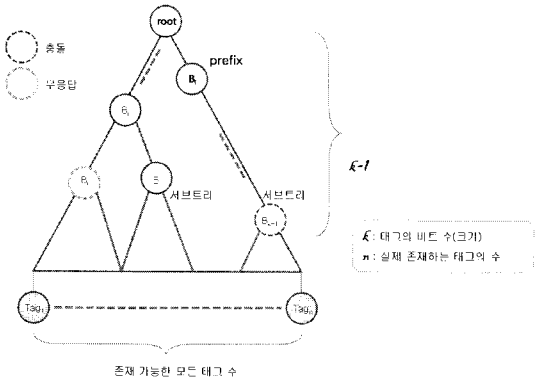

그림 8. 쿼리 트리 알고리즘 이진트리의 형태(범위 밖)
Fig. 8. Query tree algorithm binary tree(outside).

(2) Attacker가 tag의 범위 내에 존재할 경우

Attacker가 태그의 범위 내에 존재하므로 무응답의 경우와 서브트리 내에 1개만 존재하는 경우의 구분이 가능하다. 즉 그림 9에서 Tag_n가 서브트리 B_i에서 존재하는 유일한 태그라는 것을 알 수 있다는 것이다. 태그의 수를 함수식으로 표현하면 수식(8)과 같다.

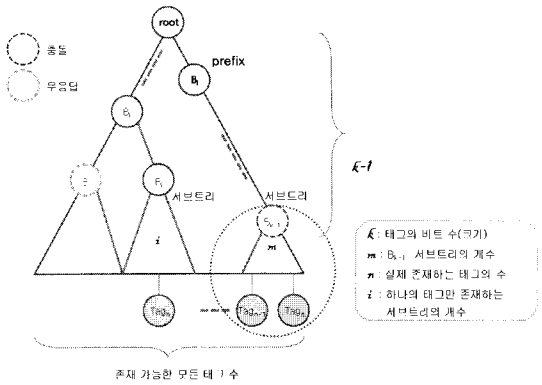

그림 9. 쿼리 트리 알고리즘 이진트리의 형태(범위 내)
Fig. 9. Query tree algorithm binary tree(inside).

$$f(k)=\{(m\times 2)+i\}+e \tag{8}$$

$$e=0$$

$$f(k)=(m\times 2)+i \tag{9}$$

$$f(k)=n \tag{10}$$

리더기가 찾아낼 수 있는 태그의 수 $f(k)$는 $k-1$에서 충돌이 발생한 경우와 하나의 태그만 존재하는 서브 트리의 수를 더한 만큼이 되며 extra tag는 존재하지 않는다. 결국 Attacker가 태그의 범위 내에 존재할 경우에는 합법적인 리더가 찾아내는 태그의 수와 동일한 태그를 Attacker가 찾아낸다는 결론이 된다.

Ⅱ장 2절의 그림 3의 사례를 적용해보면 m은 2이며, i는 1이다. 그러므로 $f(k)$는 $2\times 2+1=5$로 실재 존재하는 태그 5개를 모두 찾으므로 보안에 취약점을 보였다.

다. Advanced Query Tree 알고리즘

향상된 쿼리 트리 기반 알고리즘은 트리 워킹 알고리즘, 쿼리 트리 알고리즘과 같이 충돌 회피 알고리즘이면서, 충돌이 발생한 비트의 위치를 리더기가 검출하는 충돌 검출 기능을 함께 가지고 있다. 동작원리에서 쿼

리 트리와의 차이점은 충돌이 발생한 비트의 위치를 리더기가 검출하기 때문에 무응답의 경우가 발생하지 않는다는 것이다. 또 한 가지 차이점은 Attacker가 tag의 범위 밖에 존재할 경우 Attacker는 쿼리의 정보를 이미 가지고 있으며, 쿼리를 날리지 않고 다음 쿼리로 그냥 넘어갔을 때 넘어간 쿼리로부터 생성되는 태그의 수를 Attacker가 알고 있다는 점이다. 이것은 깊이 우선 탐색에서의 쿼리 생성 순서는 일정하기 때문이다.

수식표현을 위해 다음의 기호를 사용한다. n은 존재하는 전체 태그의 수를 나타내며 k는 태그의 비트 수(크기), m은 받은 프리픽스 중에서 비트수가 $k-1$인 프리픽스의 개수를 e는 추가로 선정된 태그의 수(overhead), i는 하나의 태그만 존재하는 서브 트리의 수를 각각 나타낸다. 또한 Attacker가 알고 있는 쿼리의 생성순서와 비교하여 예상된 쿼리를 날리지 않고 다음 쿼리로 넘어간 시점의 트리에서 생성 가능한 모든 태그의 수를 p로 표현한다. 그림 10에서 볼 수 있듯이 Attacker가 쿼리의 생성순서 a, b, x, y, z, w를 알고 있고 a, b다음의 예상 쿼리인 x가 아닌 y로 넘어간 것을 알 수 있다. 이 때 x로부터 생성가능한 모든 태그의 수를 p라 한다.

(1) Attacker가 tag의 범위 밖에 존재할 경우

쿼리 트리 알고리즘과의 차이점을 한 눈에 확인할 수 있는 부분이다. 향상된 쿼리 트리 알고리즘의 경우 Attacker가 tag의 범위 밖에 존재할 경우라도 무응답이 발생하지 않기 때문에 존재할 수 있는 모든 태그의 경우에서 무응답이 발생할 경우를 제외시켜야 한다. 수식(11)과 같이 표현할 수 있다.

$$f(k)=2^k-p \tag{11}$$

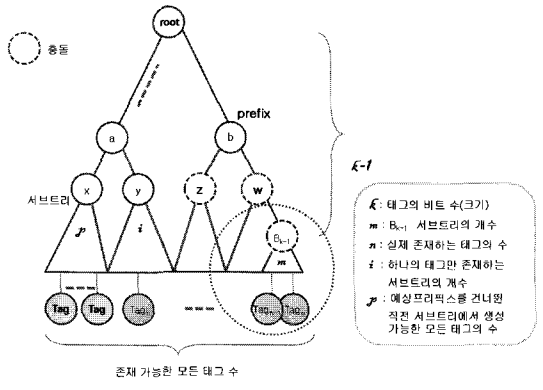

그림 10. 향상된 쿼리 트리 알고리즘 이진트리의 형태 (범위 밖)

Fig. 10. Advanced query tree algorithm binary tree(outside).

$$e=(2^k-p)-n \tag{12}$$

Ⅱ장 2절의 그림 4의 사례를 적용해보면 Attacker가 알고 있는 쿼리의 생성순서와 비교하여 예상된 쿼리를 날리지 않고 다음 쿼리로 넘어간 시점의 트리(x)에서 생성 가능한 모든 태그의 수, 즉 p는 4이다. 그러므로 Attacker는 12개의 태그를 찾으며 overhead 값은 7이 된다. 이것이 쿼리 트리 알고리즘과의 차이점이다.

(2) Attacker가 tag의 범위 내에 존재할 경우

이 경우는 앞 절에서 살펴본 쿼리 트리 알고리즘의 Attacker가 tag의 범위 내에 존재할 경우와 동일하다.

이진트리의 형태는 그림 11과 같이 쿼리 트리 알고리즘과 차이를 보이지만 Attacker가 찾는 태그의 수를 함수식으로 표현하면 위의 수식(8)~(10)과 동일하다. 리더기가 찾아낼 수 있는 태그의 수 $f(k)$는 $k-1$에서 충돌이 발생한 경우 찾은 태그에 하나의 태그만 존재하는 서브 트리의 수를 더한 만큼이 되며 extra tag는 존재하지 않는다. 결국 Attacker가 태그의 범위 내에 존재할 경우에는 합법적인 리더가 찾아내는 태그의 수와 동일한 태그를 Attacker가 찾아낸다는 결론이 된다.

Ⅱ장 2절의 그림 4의 사례를 적용해보면 m은 2이며, i는 1이다. 그러므로 $f(k)$는 $2\times2+1=5$로 실제 존재하는 태그 5개를 모두 찾아 보안에 취약함을 알 수 있다.

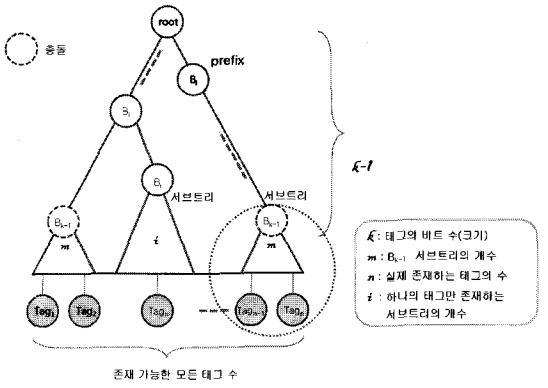

그림 11. 향상된 쿼리 트리 알고리즘 이진트리의 형태 (범위 내)

Fig. 11. Advanced query tree algorithm binary tree(inside).

라. 결과비교

(1) Attacker가 tag의 범위 밖에 존재할 경우

본문을 통해 살펴본바와 같이 각 알고리즘들은 각각의 다른 overhead값 (e)을 가지며 우리는 그 값을 통해 견고성을 비교할 수 있다. overhead 값이 크다는 것은

extra tag가 많다는 것으로 견고성이 높다는 것이며, overhead 값이 작다는 것은 반대로 extra tag가 적다는 것으로 견고성이 취약하다는 것이다.

앞서 살펴본 예제와 두 개의 예제를 추가로 결과비교를 해보고자 한다. Ⅱ장 2절의 예제에서 overhead값은 각각 트리 워킹 알고리즘에서 1개, 쿼리 트리 알고리즘에서 11개, 향상된 쿼리 트리 알고리즘에서 7개로 쿼리 트리 알고리즘이 Attacker로부터 가장 보안에 견고함을 보였다.

두 번째 예로 Attacker가 tag의 범위 밖에 존재하며 리더의 인식범위에 8비트 크기의 다중태그 6개 ('00010001', '00010101', '00110101', '10001010', '11001011', '11010010')가 존재한다고 할 때 각 알고리즘에 대한 overhead값은 트리 워킹 알고리즘에서 0개, 쿼리 트리 알고리즘에서 250개, 향상된 쿼리 트리 알고리즘에서 136개이다.

세 번째 예로 Attacker가 tag의 범위 밖에 존재하며 리더의 인식범위에 8비트 크기의 다중태그 8개 ('00010001', '00010101', '00100010', '00110001', '10001100', '10001101', '11010010', '11010100')가 존재한다고 할 때 각 알고리즘에 대한 overhead값은 트리 워킹 알고리즘에서 1개, 쿼리 트리 알고리즘에서 248개, 향상된 쿼리 트리 알고리즘에서 216개이다.

표 1에서 보는 바와 같이 모든 경우에서 트리 워킹 알고리즘의 경우 extra tag가 거의 없을 정도로 근접한 태그를 찾아냈다. 이것은 그만큼 정확한 정보의 태그를 찾는다는 것이므로 보안에 취약하다는 것인 반면 쿼리 트리 알고리즘의 경우 존재하는 태그의 최소 2배 이상 많게는 42배가 넘는 extra tag를 확인할 수 있다. 그만큼 어떠한 태그가 존재하는지 알아내지 못할 만큼 보안에 강하다는 것을 알 수 있는 결과이다. 보안 견고성을

표 1. 각 알고리즘의 extra tag 수 비교
Table 1. Extra tag number of each algorithm.

(단위 : 개)

알고리즘 / Finding tag / Extra tag / 적용예제	트리 워킹 (TW)		쿼리 트리 (QT)		향상된 쿼리 트리(AQT)		비고
	f(k)	e	f(k)	e	f(k)	e	
예제1 (4비트_5개)	6	1	16	11	12	7	
예제2 (8비트_6개)	6	0	256	250	142	136	
예제3 (8비트_8개)	9	1	256	248	224	216	

간단히 부등식으로 표현해보면 수식(13)과 같다.

$$QT \geq AQT > TW \tag{13}$$

(2) Attacker가 tag의 범위 내에 존재할 경우

Attacker가 tag의 범위 내에 있을 경우 합법적인 리더와 동일하게 모든 태그를 찾아낼 수 있으므로 모든 알고리즘은 외부로부터의 Overhearing에 매우 취약함을 보였다.

Ⅳ. 결 론

본 논문에서는 유비쿼터스 컴퓨팅 기술의 핵심기술인 RFID 시스템에서 Tag의 정보를 감지하기 위해 제안된 Tree 기반 알고리즘의 보안 견고성에 대하여 살펴보았다. 모든 알고리즘의 경우 Attacker의 위치가 tag의 범위 내에 존재하는지의 여부는 각 알고리즘의 보안 견고성을 판단하는데 매우 큰 영향을 미친다. Attacker가 tag의 범위 내에 있을 경우 합법적인 리더와 동일하게 모든 태그를 찾아낼 수 있으므로 모든 알고리즘은 외부로부터의 Overhearing에 매우 취약함을 알 수 있다. 하지만 Attacker가 tag의 범위 밖에 존재할 경우 상대적으로 많은 수의 extra tag(실제 존재하지 않는 태그)를 인식하게 되는 향상된 쿼리 트리 알고리즘과 쿼리 트리 알고리즘이 트리 워킹 알고리즘에 비해 견고함을 확인 할 수 있다. 만약, 인식 범위에 분포 되어있는 태그의 수가 적을 경우 각 알고리즘이 인식하는 extra tag의 수는 그 차이에 있어서 그리 큰 것이 아닐 수 있다. 하지만 RFID의 특성 상 대부분의 응용 분야에서 리더가 인식해야하는 태그의 수는 적게는 수십, 많게는 수천 개에 달하므로 이 때 발생하는 extra tag의 수는 각 알고리즘의 보안 견고성을 확실하게 보여준다.

향후 논문에서는 다루었던 트리 기반 충돌 방지 알고리즘과 슬롯 알로하(Slotted ALOHA) 기반 알고리즘의 보안 견고성에 대하여 비교 및 분석하겠다.

참 고 문 헌

[1] Ari Juels, Ronald L Rivest, Michael Szydlo, "The Blocker Tag:Selective Blocking of RFID Tags for Consumer Privacy", Proceedings of the 10th ACM conference of Computer and communication security, ISBN:1-58113-738-9,

PP.103-111, 2003.
[2] Ching Law, Kayi Lee and Kai-Yeung Sju, "Efficient Memoryless Protocol for Tag Identification", Proceedings of the 4th International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communication, ACM, pp.75-84, August, 2003.
[3] 김정숙, 김천식, 윤은준, 홍유식, "RFID와 TCP/IP를 활용한 원격보안 출입 제어 시스템", 전자공학회논문지 제45권 CI편 제6호, 2008년 11월
[4] 남택용, 장종수, 손승원, "유비쿼터스 환경에서의 개인정보 보호기술", 전자통신동향분석 제20권 제1호, 2005년 2월
[5] 서현곤, 한재일 "RFID시스템에서 충돌 트리 기반 충돌 방지 알고리즘, 한국정보과학회논문지 : 정보통신, 제34권, 제4호, 2006년
[6] D. Brock, "The Electronic Product Code - A Naming Sche me for Physical Objects", Auto-ID White Paper, January 2001.
[7] H, Knospe and H. Pobl, "RFID Security", Information Security Technical Report, vol. 9, no. 4, pp. 39-50, Elsevier, 2004.
[8] S. Sarma, S. Wies, and D. Engels, "Radio-Frequency identification : security Risks and Challenges", In Crypto bytes, vol. 6, no. 1, pp. 2-9, RSA Laboratories, Spring 2003.
[9] R. Anderson and M. Kuhn, "Low cost attacks on tamper resistant devices", In Processing of the International Workshop on Security Protocols - IWSP, vol. 1361 of LNCS, PP. 125-135, April 1997.
[10] R. Revest, "Approaches to RFID Privacy", RSA Japan Coference, 2003.
[11] Auto-Id Center, "860㎒ ~ 960㎒ class 1 radio frequency identification interface specification proposed recommendation version 1.0.0", Technical Report MIT AUTOID-TR-007, November 2002.
[12] S. Sarma, S. Weis, and D. Engels, "RFID Systems and Security and Provacy Inplications", In Proceeding of the Cryptographic Hardware and Embedded Systems - CHES 2002, vol. 2523 of LNCS, pp. 454-469, August 2002.
[13] H. Vogt, "Efficient Object Identification with Passive RFID Tags", Intn'l Conference on Pervasive Computing, Zurich, 2002.
[14] 최원준, 노병희, 유승화, 오영철, "랜덤화된 트리워킹 알고리즘에서의 RFID 태그 보안을 위한 백워드 채널 보호 방식", 한국통신학회논문지, vol.30, no.5c, 2005년 5월
[15] 나종민, "트리-워킹 방식 기반의 RFID 시스템 스푸핑 방어 알고리즘", 광운대학교 대학원 컴퓨터과학과 석사학위 논문, 2005년 12월Circuits and Systems for Video Tech., Vol. 8, no. 3, pp. 345-357, June 1998.

저 자 소 개

서 현 곤(정회원)
1994년 경성대학교 이학사
1996년 경성대학교 이학석사
2004년 영남대학교 공학박사
1994년~1997년 티센크루프동양 엘리베이터(주) 기술연구소 주임연구원
2001년~2003년 대구대학교 정보통신공학부 BK21교수
2004년~2005년 영남대학교 컴퓨터공학과 강의전담교수
2005년~현재 한라대학교 정보통신방송공학부 교수
<주관심분야 : 6LoWPAN, RFID/USN, Embedded System>

김 향 미(학생회원)
1996년 충주대학교 공학사
2009년 한라대학교 정보통신공학 공학석사
2006년~2009년 강원원주대학교 정보통신공학과 조교
<주관심분야 : 센서 네트워크, RFID/USN, HCI>