RFID 시스템에서의 인증 및 위치추적 방지 모델*

김진묵** · 유황빈**

요 약

본 논문에서는 차세대 패킷 네트워크에서의 성능 품질을 개선시키기 위한 하나의 접근 방안으로 패킷 지연에 대한 절대적 차별화 기능을 제공하는 알고리듬을 제안한다. 제안된 알고리듬은 임의 시간 구간에 도착될 입력 트래픽을 예측하고 이를 기반으로 지연 제어 기능을 수행한 후 실제로 입력된 트래픽을 측정하여 예측 오차를 도출하고 이를 다음 시간 구간에서 보상하는 것을 특징으로 한다. 제안된 방식은 매 시간 구간마다 지속적으로 예측 오차를 보상함으로서 예측 편차가 높은 버스트 트래픽에 대하여 특히 우수한 성능을 제공한다. 모의 실험을 통해 제안된 방식은 절대적 성능 지표를 충족하고 기존 방식에 비해 버스트 트래픽에 대하여 우수한 적응성을 제공하는 것을 확인한다.

An Certification and a Location Tracing Protect Model on RFID*

Jin Mook Kim** · Hwang Bin Ryou**

ABSTRACT

RFID System has an advantage that it need not touch an objects for identification of many objects. Because it is working through wireless communication. Also, So many objects can be identified with RFID System at once. However, although RFID System has convenience like above, it has serious privacy concern at the same time. If RFID System is working with an target object through wireless communication, other objects will respond to RFID System signal as well as a target object. Hence, RFID System can be easily exposed user privacy by attacker. In this paper, We propose RFID system authentication model in order to protecting user privacy and traking. Proposed RFID system is operating that not only server authenticate RFID reader but also RFID reader and tag authenticate mutually by using symetric cryptography that operating with tiny and simple processing.

Key words : RFID protection, Authentication, Trace avoidience

* 이 논문은 2005년도 광운대학교 교내연구비에 의해 연구되었음.

** 광운대학교 컴퓨터소프트웨어학과

1. 서 론

유비쿼터스 컴퓨팅 환경에서는 모든 사물들에 태그를 부착하고 비-접촉방식으로 대량으로 사물 인식이 가능한 RFID 시스템에 적용될 것으로 예상된다[10].

그리고 사물에 대한 기본정보를 습득하기 위해서 기존과 달리 무선 주파수를 이용하기 때문에 정보를 습득할 수 있는 리더 장치를 소유하기만 하면 누구나 사물에 대한 기본 정보를 습득할 수 있다. 그리고 사물에 대한 습득한 정보들을 지속적으로 축적하여 사물에 대한 위치 정보를 추적할 수도 있다는 심각한 문제점을 내포하고 있다[1].

하지만 이러한 문제점을 해결하기 위해 기존의 유선 통신 방식에서 제안된 보안 및 프라이버시 해결책들을 RFID 시스템에 직접적으로 적용하기도 어려움이 따른다. 이는 RFID 시스템에서 사용되는 통신의 주체인 태그와 리더가 매우 낮은 처리 능력과 저장 능력을 가지고 있기 때문이다[2-4].

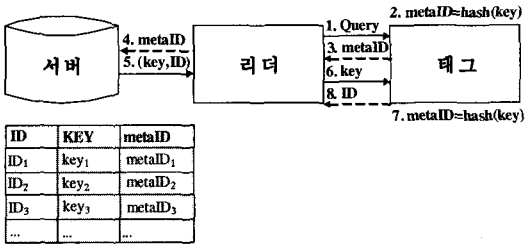
이에 본 논문에서는 RFID 시스템에서 태그가 가지고 있는 하드웨어적인 제약조건과 리더와의 통신 과정에서 무선 통신방식을 사용함으로 인한 어려움들을 감안하여 단순한 처리방식과 적은 연산 횟수에도 불구하고 태그와 리더 사이에서의 통신에서 인증 서비스를 제공하고, 이를 통해 태그에 대한 위치 추적을 방지할 수 있는 모델을 제안하고자 한다. 제안하는 모델에 대한 태그와 리더에 대한 시뮬레이션을 통해 처리 성능을 측정해 봄으로서 현실 적용 가능성을 가늠해 보고자 한다.

2. 관련 연구

기존에 RFID 시스템에 대한 익명성 및 위치 추적에 대한 문제를 해결하기 위한 대표적인 해결책으로 MIT의 해쉬-락 기법과 NTT의 해쉬-체인 기법이 있다[5].

2.1 MIT의 해쉬-락 기법

MIT에서는 태그와 리더 사이에의 통신에 태그에 대한 익명성을 보장하고 인증된 리더에 대해서만 태그에 대한 정보를 제공하기 위해서 공개키 방식의 해쉬함수를 기반으로 하는 해쉬-락 기법을 제안하였다.



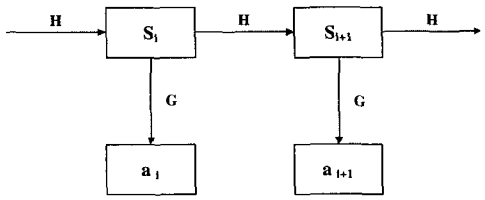
ID	KEY	metaID
ID_1	key_1	$metaID_1$
ID_2	key_2	$metaID_2$
ID_3	key_3	$metaID_3$
...	...	...

(그림 1) MIT의 해쉬-락 기법

하지만 해쉬-락 기법은 익명성을 제공할 수 있으나 반복적인 질의에 매번 같은 metaID를 제공한다는 것과 공개키 방식의 해쉬함수를 기반으로 함으로 적은 자원을 갖는 태그에서 실현가능성이 적다는 단점을 갖고 있다.

2.2 NTT의 해쉬-체인 기법

NTT의 Ohkubo는 기존의 MIT에서 제안한 해쉬-락 기법의 단점들을 고려하여 익명성과 위치 추적에 대한 해결책으로 해쉬-체인 기법을 제안하였다.



(그림 2) NTT의 해쉬-체인 기법

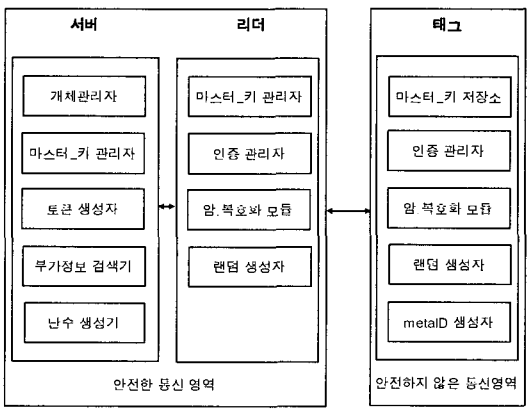
하지만 해쉬-체인 기법에서도 리더의 질의에

대해 항상 새로운 익명성이 보장되는 임시 식별자를 응답함으로써 위치 추적에 대한 문제는 해결할 수 있으나 공개키 기반의 해쉬함수를 기반으로 함으로써 낮은 하드웨어 제약 조건을 갖는 태그 환경에 적용하는데 어려움은 여전히 남게 된다.

3. 인증 및 위치추적 방지 모델

3.1 인증 및 위치추적 방지 모델 구조

본 논문에서 제안하는 인증 및 위치 추적 방지를 위한 모델은 통신을 위해서 사전에 리더와 태그에 대한 인증을 확인하고 이를 토대로 상호간에 통신에 사용할 세션키를 생성한 후에 암호화된 통신을 수행할 수 있다. (그림 3)은 제안 모델의 구조를 나타내고 있다.



(그림 3) 제안하는 인증 및 위치추적 방지 모델

제안 모델에서 태그는 수동형 태그의 표준으로 인정받고 있는 EPC-96의 구조와 유사하다. 추가로 태그와 리더 사이에서의 통신에서 사용할 마스터-키와 리더와 태그의 인증을 위한 인증관리자, 암.복호화 모듈, 익명성을 보장하기 위해서 랜덤 생성자와 임시 식별자인 metaID 생성자 모듈을 탑재하고 있다.

태그와 통신을 수행하는 리더는 일반적으로 무선 통신이 가능하고 태그와의 통신에서 인증을 위한 인증 관리자, 암.복호화 모듈, 태그와 세션키를 생성하기 위한 랜덤 생성자 모듈을 포함하고 있다.

재안 모델에서 서버는 기본적으로 태그와 리더에 대한 정보를 저장하고 관리하기 위한 개체 관리자, 리더와 태그의 마스터-키를 관리하기 위한 마스터-키 관리자, 리더와 태그의 통신에서 인증을 위한 토큰 생성자, 태그에 대한 부가 정보를 검색하기 위한 부가정보 검색기, 난수 생성기로 구성된다.

3.2 제안 모델의 동작과정

제안한 인증 및 위치추적 방지를 위한 모델의 세부 동작 과정을 살펴보기 위하여 흐름도와 처리 절차에서 사용하는 기호와 메시지들을 <표 1>과 <표 2>로 정리하여 나타낸다.

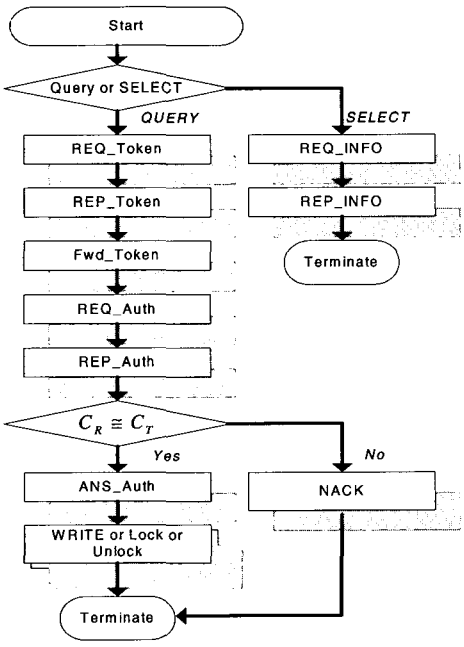
〈표 1〉 제안모델에서 사용하는 기호

기호	설명
‖	변수들을 연접하기 위한 기호
⊕	XOR 연산을 위한 기호
≅	수식의 동일성여부 판별 기호
S	서버를 지칭하는 기호
EPC	AutoID 센터의 태그 식별기호
RN_R	리더가 생성한 임의의 난수
RN_T	태그가 생성한 임의의 난수
RN_S	서버가 생성한 임의의 난수
K_R	리더가 소유한 마스터-키
K_T	태그가 소유한 마스터-키
meta	EPC 대응하는 임시 식별기호
C_R	서버가 생성해 준 리더 서명값
C_T	서버가 생성해 준 태그 서명값
$Auth_R$	리더의 인증 여부 확인 정보
$Auth_T$	태그의 인증 여부 확인 정보
E(K;M)	K를 사용하여 M을 암호화
D(K;M)	K를 사용하여 M을 복호화
CBC(K;M)	K를 사용하여 M을 CBC 모드로 MAC 생성

〈표 2〉 제안모델에서 사용하는 메시지

메시지	설명
SELECT	일반 통신을 요청
QUERY	부가 통신을 요청
meta	태그 임시 식별자
REP	일반 통신요청에 대한 응답
REQ_INFO	서버에 정보검색 요청
REP_INFO	서버의 정보검색 결과 응답
REQ_Token	서버에 토큰 생성 요청
REP_Token	생성된 토큰에 대한 응답
REQ_Auth	리더 인증확인 요청
REP_Auth	리더 인증확인 요청 응답
ANS_Auth	태그 인증확인 응답
ACK	접근 허락
NACK	통신 취소
Terminate	통신 종료
Write	태그에 기록
Lock/Unlock	태그 잠금/ 잠금해제

(그림 4)는 제안한 모델에서의 전체적인 동작과정을 흐름도로 나타낸 것이다.



(그림 4) 제안모델의 전체동작과정

(그림 4)에서 나타낸 것처럼 RFID 시스템의 구성요소인 리더와 태그 사이에서 리더의 통신 요청으로부터 시스템은 동작하게 된다.

태그는 리더의 통신요청에 대해 통신 요청을 위한 메시지를 QUERY와 SELECT로 구분한 후, 일반적인 통신을 요청하는 SELECT 메시지를 전달받은 경우에는 태그의 임시 식별자(meta)만을 전송하고 잠금 상태를 유지한다[5].

리더로부터의 통신 요청을 위한 메시지가 QUERY인 경우에는 리더가 태그에게 부가적인 정보를 요청하거나 태그에 액세스를 요청하는 것으로, 사전에 리더와 태그 사이에서 인증 과정을 필요로 한다. 이를 위해서 서버가 리더와 태그에게 토큰을 생성해 전달하고 이를 이용해 인증과정을 수행한다. 이는 Tom Leighton과 Micali에 의해 제안된 비밀키 방식을 이용한 기법으로 이미 안전성이 입증된 바 있다[6].

리더와 태그 사이에서 상호 인증을 확인하고 나면 태그는 잠금 상태를 해제하고 리더의 액세스 요청에 대한 처리를 수행하고 정상적으로 통신을 종료하게 된다.

3.3 인증 및 위치추적 방지 처리절차

3.3.1 사전 등록

RFID 시스템에서 인증 및 위치추적 방지를 위해서 생산 단계에서 부여된 태그와 리더에 대한 생산정보와 마스터-키를 서버에 등록한다.

이때 서버에 저장되는 리더와 태그의 정보를 저장하기 위해서 정의하여 사용하는 데이터 구조를 (그림 5)와 같이 나타낸다.

사전에 서버의 데이터베이스에 태그와 리더의 생산 단계에서 부여된 정보들을 저장하기 위한 테이블을 생성해 두고, 하위의 세부절차에서 발생하는 데이터들을 추가적으로 변경할 수 있도록 한다.

또한 데이터베이스에 태그에 기록된 생산번호

와 매핑되는 사물에 대한 정보들을 테이블로 생성하여 기록해 두고 이를 리더의 요청에 따라서 검색하여 응답하도록 한다.

생산번호	EPC	비밀키	공유키	인증값
3922	96bit	128bit	128bit	128bit
3904	96bit	128bit	128bit	128bit
...	...	...	...	...
0370	96bit	128bit	128bit	128bit

태그_테이블

생산번호	유저번호	비밀키	공유키	인증값
2230	128bit	128bit	128bit	128bit
2304	128bit	128bit	128bit	128bit
...	...	...	...	...
3292	128bit	128bit	128bit	128bit

리더_테이블

(그림 5) 리더와 태그의 사전등록 정보

3.3.2 일반통신 요청

리더는 SELECT 메시지를 사용해 태그에게 식별만을 위한 임시 식별자를 요청할 수 있다. (그림 6)에서 이에 관한 처리절차를 나타낸다.

내부 처리절차들을 살펴보면 다음과 같다.

① R => T : $SELECT = id_R \,||\, RN_R$

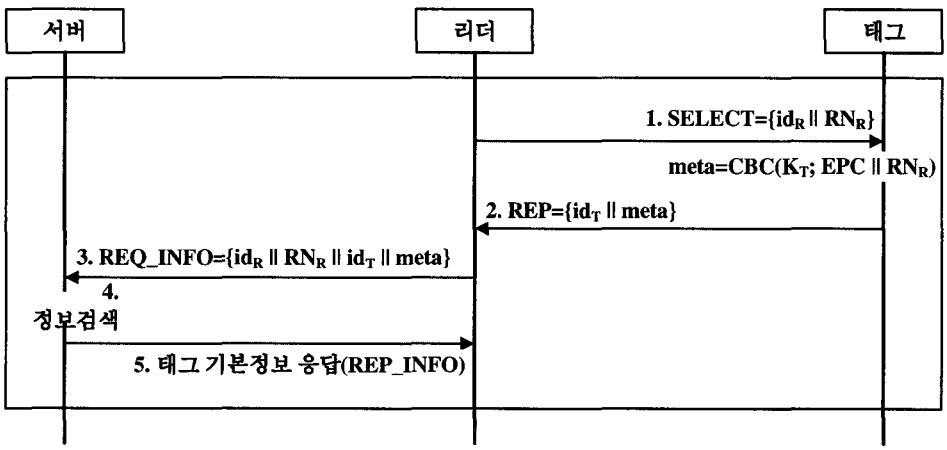
리더가 일반적인 통신을 요청하기 위해서 생산번호(id_R)와 리더가 생성한 임의의 난수(RN_R)을 연접하여 SELECT 메시지를 생성해 태그에게 전달한다.

② T => R : $REP = id_T \,||\, meta$

태그는 리더의 SELECT 메시지에 대한 응답으로 태그에 대한 임시 식별자인 meta를 생성해 전달한다. 이를 통해 리더는 태그를 부착한 사물에 대한 식별이 가능하다. 하지만 새로운 SELECT 메시지마다 항상 새로운 meta를 생성하여 응답함으로써, 리더가 태그에 대한 식별자를 축적하여 태그가 부착된 사물에 대한 위치 추적이 어렵게 한다.

$$meta = CBC(K_T;\ EPC \,||\, RN_R)$$

임시 식별자인 meta는 수식과 같이 CBC 모드의 RC5 대칭키 암호 알고리즘을 통해 MAC을 생성하게 된다. 이때 태그에 대한 고유한 식별을 하면서 리더의 요청마다 새로운 식별자를 생성하기 위해서 태그의 고유정보인 EPC와 리더가 생성한 임의의 난수 RN_R을 연접한 후, 태그의 고유 비밀키로 MAC 처리한다. 이렇게 함으로서 기존의 공개키 방식의 해쉬함수보다 빠르면서도 유사한 안



(그림 6) 일반통신 요청에 대한 처리절차

전성을 보장할 수 있다.

③ R => S : $REQ\_INFO = `id_R || RN_R || id_R || meta'$

리더는 태그로부터 전달받은 임시 식별자 meta를 이용해 서버에게 태그에 관한 일반적인 정보를 요청하기 위해서 REQ_INFO를 생성하여 전달한다.

④ S : DB 검색

서버는 전달받은 REQ_INFO를 사용하여 데이터베이스를 검색하여 리더가 원하는 태그에 대한 정보를 검색한다.

이때 서버가 태그에 대한 인증을 위해서 전송받은 meta와 자신의 데이터베이스를 검색하여 생성한 meta'을 비교하여 동일한 경우에만 태그를 인증하고 이에 대한 정보를 리더에게 전송할 수 있다.

⑤ S => R : *RES_INFO*

서버는 태그에 대한 부가가치가 없는 일반적인 식별정보들을 연접하여 RES_INFO를 생성하고 이를 리더에게 전달한다.

3.3.3 토큰 생성

태그가 부착된 사물에 대한 부가정보를 요청하거나 리더가 태그에 직접적인 액세스를 요청하는 경우, 인증여부를 확인하여야 한다. 이를 위해서 토큰을 서버로부터 생성하여 전달받아야 한다. 자세한 처리절차는 (그림 7)에 나타낸다.

① R => T : $QUERY = `id_R || RN_R'$

리더가 부가적인 통신을 요청하기 위해서 생산번호(id_R)와 리더가 생성한 임의의 난수(RN_R)을 연접하여 QUERY 메시지를 생성하여 태그에게 전달한다.

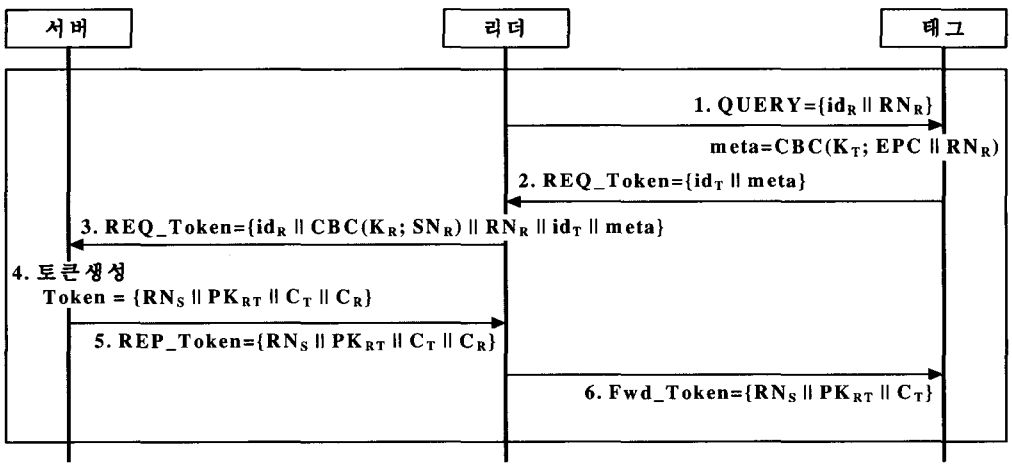
② T => R : $REQ\_Token = `id_T || meta'$

태그는 리더의 QUERY 메시지에 대해 리더와 태그 사이에서 사용할 공개키와 인증을 위한 정보들을 필요로 하게 된다. 이에 자신의 생산번호와 임시 식별자를 연접하여 REQ_Token 메시지를 생성한 후 리더에게 전달한다. 이를 통해서 태그가 리더의 액세스 허용 여부와 인증 요청을 알린다.

③ R => S : REQ_Token

$= `id_R || CBC(K_R; SN_R) || RN_R || id_R || meta'$

리더는 태그가 자신에 대한 인증을 요청하는 REQ_Token에 리더의 생산번호와 식별번호(SN_R)을 이용해 MAC을 생성하여 태그로부터 전달받은



(그림 7) 리더와 태그의 인증을 위한 토큰 생성 및 분배절차

REQ_Token에 연접하여 서버에게 전달함으로서 토큰을 생성해 줄 것을 요청한다.

④ S : Token 생성

서버는 리더의 요청에 따라 토큰을 생성한다. 서버의 임의의 난수와 리더와 태그 사이에서 사용할 공개키, 리더와 태그에 대한 각각의 인증값들을 연접하여 토큰을 생성하게 된다.

$$Token = `RN_S \,||\, PK_{RT} \,||\, C_T \,||\, C_R`$$

서버로부터 생성된 토큰에서 리더와 태그 사이에서 사용할 공유 비밀키는 다음에 나타내는 수식과 같이 생성된다.

$$PK_{RT} = KM_R \oplus KM_T$$
$$KM_R = CBC(K_R; RN_S)$$
$$KM_T = CBC(K_{R;RN_s})$$

그리고 각각의 태그와 리더에 대해 서버가 생성해 준 인증값 C_T와 C_R은 아래의 수식에 의해 생성되고 인증확인을 위해 사용한다.

$$C_R = CBC(K_R; S \,||\, id_R \,||\, meta \,||\, KM_T)$$
$$C_T = CBC(K_T; S \,||\, meta \,||\, id_R \,||\, KM_R)$$

⑤ S => R :

$REP\_Token = `RN_S \,||\, PK_{RT} \,||\, C_T \,||\, C_R`$

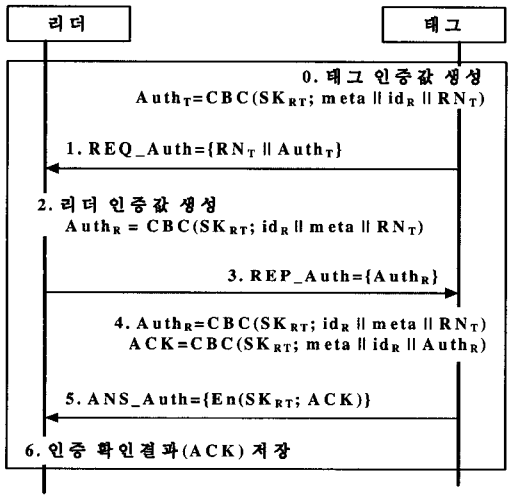
서버는 생성한 토큰을 리더에게 REP_Token 메시지로 전달한다.

⑥ R => T : $Fwd\_Token = `RN_S \,||\, PK_{RT} \,||\, C_T`$

서버로부터 토큰을 전달받은 리더는 전달받은 토큰 정보를 저장하고, 자신의 인증값인 C_T를 제외한 나머지 토큰값을 태그에게 전달하기 위해서 Fwd_Token 메시지를 전송한다.

3.3.4 인증 확인

리더와 태그는 상대방에 대한 인증여부를 검증하기 위해서 서버로부터 생성하여 전달받은 각각의 인증값과 상대방이 생성해 전달한 인증값의 동일성 여부를 확인하여 인증 여부를 결정하게 된다. 이를 (그림 8)과 같이 나타낸다.



(그림 8) 인증 확인 절차

리더와 태그는 서버가 생성해 전달한 토큰을 사용하여 상호 인증을 확인을 위하여 각각의 Auth 값을 생성하여 전달한다. 그리고 이에 대한 검증을 수행하고, 이에 대한 결과를 ACK 혹은 NACK로 응답하게 된다.

① T : $Auth_T$ 생성

$Auth_T = CBC(SK_{RT};\ meta \,||\, id_T \,||\, RN_T)$

태그는 리더와 상호간에 인증 가능성 여부를 확인하기 위해서 자신에 대한 고유한 인증값 $Auth_T$를 생성하여 전달한다. 이때 공유 비밀키(PK_{RT})로부터 파생한 세션키(SK_{RT})를 사용해 MAC을 생성하기 때문에 사전에 서버로부터 전달받은 토큰을 가지고 있지 않은 리더는 이에 대해 해독이 불가능하다.

② T => R : $REQ\_Auth = `RN_T \,||\, Auth_T`$

태그는 리더에게 REQ_Auth 메시지를 전송함

으로써 자신의 고유한 인증값을 전달하고, 이를 이용해 리더의 고유한 인증값을 생성하여 전달해 줄 것을 요청한다.

③ R : $Auth_R$ 생성

$Auth_R = CBC(SK_{RT}; id_R \,||\, meta \,||\, RN_T)$

리더는 태그에 대한 자신의 고유한 인증값인 AuthR을 생성하여 전달함으로서 상호 인증을 얻고자 한다.

④ R => T : $RES\_Auth = `Auth_R'$

리더는 태그와의 상호 인증을 확인하기 위해서 자신이 생성한 AuthR을 전달하기 위해서 RES_Auth 메시지를 전송한다.

⑤ T : 리더 인증값 검증

태그는 리더로부터 전달받은 AuthR과 자신이 공유 정보를 이용하여 생성한 Auth'R의 동일성을 비교한다. 만약 동일하다면 리더와 태그 사이에는 상호 인증이 성립하는 것으로 응답메시지로 ACK를 생성하고, 그렇지 않을 경우에는 NACK를 생성한다. 태그의 응답메시지 ACK는 다음의 수식과 같이 생성한다.

$$ACK = CBC(SK_{RT};\ meta \,||\, id_R \,||\, Auth_R)$$

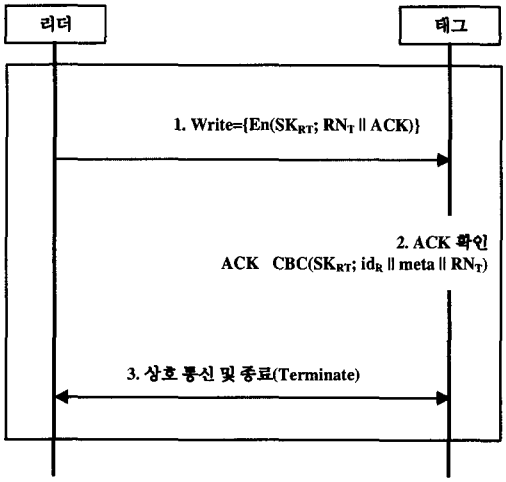
⑥ T => R : $ANS\_Auth = `E(SK_{RT};\ ACK)'$

마지막으로 서버는 인증이 확인된 경우에 대해서는 ACK를 전달함으로서 상호 인증이 성립함을 알려준다.

3.3.5 인증된 부가통신

리더와 태그 사이에서 상호 인증이 확인된 경우라고 하더라도 리더가 태그를 직접 액세스 하려고 하는 경우에 불의의 문제가 발생할 소지가 있다.

(그림 9)는 상호 인증을 거친 태그와 리더는 ACK를 저장하고 있으므로 이를 포함하여 직접 액세스를 요청하는 메시지를 생성하여 전달할 수 있음을 나타내고 있다.



(그림 9) 인증된 부가통신 절차

① R => T : Write 메시지

$Write = `E(SK_{RT};\ RN_T \,||\, ACK)'$

리더는 태그에 직접 데이터를 변경하기 위해 쓰기를 요청하는 Write 메시지를 전송한다. 이때 상호 인증이 이루어졌음을 증명하는 ACK와 태그가 생성한 임의의 난수를 함께 세션키로 암호화하여 전달한다.

② T : ACK'을 생성

$ACK' = CBC(SK_{RT}; id_R \,||\, meta \,||\, RN_T)$

태그는 리더로부터 전달받은 ACK의 진위여부를 확인하기 위하여 MAC을 생성한다.

그리고 리더로부터 전달받은 ACK와 자신이 생성한 ACK'의 동일성 여부를 판별한다.

③ R <==> T : Terminate 전달

리더와 태그는 언제라도 정상적이거나 그렇지 않을 때라도 통신을 종료하는 시점에서 이를 상대방에게 알리기 위해서 Terminate 메시지를 전송한다. 이를 통해서 상호간에 설정된 세션은 파괴되고 태그는 잠금 상태가 된다.

4. 시뮬레이션

지금까지 RFID 시스템에서 서버와 리더나 태그에 대한 인증, 그리고 리더와 태그 사이에서의 인증을 보장할 수 있고, 태그에 대한 식별은 가능하지만 위치 추적을 할 수 없도록 할 수 있도록 제안한 모델에 대해 기술하였다.

현존하는 RFID 시스템은 현실적으로 본 논문에서 제안하는 바와 같이 태그 영역에 확장된 필터 영역을 가지고 있지 않다. 또한 추가적으로 암호화나 복호화를 위한 처리모듈을 탑재하기에 부족한 자료저장 공간을 가지고 있다.

그리고 처리장치도 탑재하고 있지 못한 것이 일반적인 수동형 태그의 실정이다. 그러므로 논문에서 제안한 실험을 위해서 태그와 리더에 대한 시뮬레이터를 구현하도록 하였다.

헤더	필터	생산자번호	품목번호	일련번호
8 bit	3 bit	28 bit	24 bit	36 bit

구분	설 명
000	일반적인 모든 통신 목적으로 사용
001	예약된 필드
010	예약된 필드
011	예약된 필드
100	토큰 생성을 위한 목적으로 사용
101	예약된 필드
110	설명
111	인증 확인을 위한 목적으로 사용

(그림 10) 데이터구조 및 필터 규격

(그림 10)은 제안한 시뮬레이터에서 리더와 태그 사이에서 통신을 위해서 전달하기 위한 EPC 코드를 바탕으로 생성한 데이터구조를 나타내고 있다. 이때 추가적으로 필터 영역을 두어 통신에 대한 메시지들을 분류하고 이에 대한 처리를 효과적으로 수행할 수 있도록 하였다.

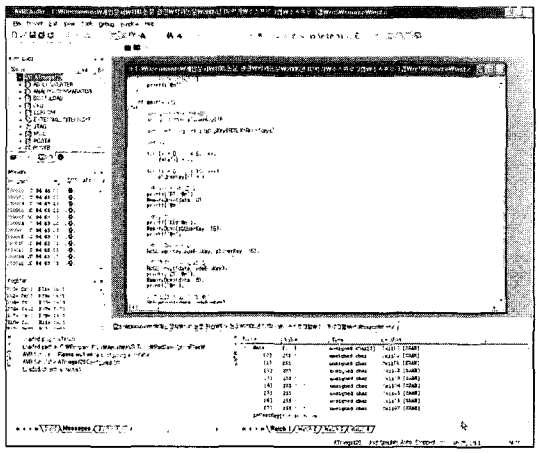
4.1 시뮬레이터 구현환경

<표 3>은 태그와 리더에 대한 시뮬레이터를 구현하기 위한 환경에 대해 기술하고 있다.

〈표 3〉 시뮬레이터 구현환경

구분	구현 환경
태그	시뮬레이터 • Programmers Notepad 2.0.5.48 • WINAVR 컴파일러 20050214버전 • AVR-Libc 1.2.3 라이브러리 • AVR STUDIO 디버거 동작환경 • Windows XP 탑재된 펜티엄 4 데스크탑 환경에서 동작하는 daemon
리더 & 서버	운영체제 : Windows XP SP2 구현환경 : VC++ 6.0 MFC

태그 시뮬레이터는 Atmega128 칩을 기준으로 8 bit 처리장치와 4KB의 저장공간을 탑재하고 있음을 가정하여 시뮬레이터를 구성하였다.



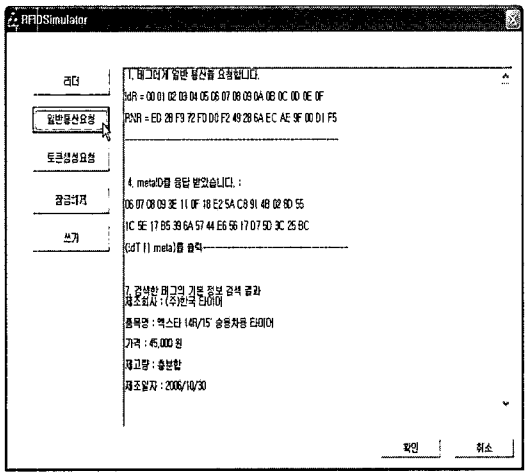
(그림 11) 태그 시뮬레이터 디버그 화면

(그림 11)은 태그에 대한 시뮬레이터를 구현하

고, 태그 시뮬레이터가 올바르게 동작하는지 여부를 체크하기 위해서 디버깅 작업을 수행하는 화면을 나타내고 있다.

4.2 시뮬레이터 구현 결과

리더에 대한 내부 모듈들을 탑재하고 제안한 모델처럼 동작하는 리더 시뮬레이터를 구현한 결과를 (그림 12)가 나타낸다.

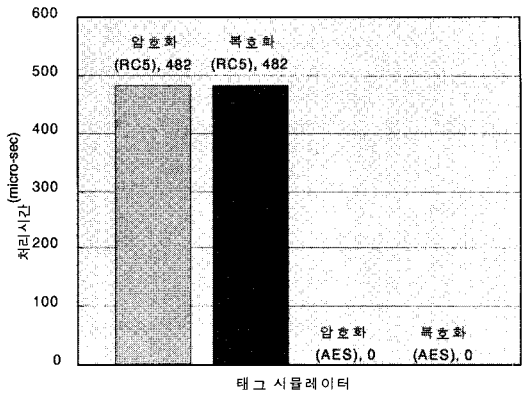


(그림 12) 리더 시뮬레이션 구현 결과

5. 성능평가 및 결론

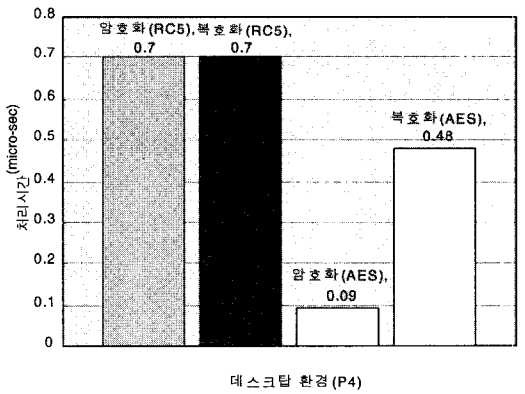
본 논문에서 제안한 인증 및 위치추적 방지를 위한 모델에 대해 시뮬레이터를 구현하고 이를 통하여 데이터를 처리하는 과정에 대한 처리 시간을 측정하였다.

1Kbyte 크기의 데이터를 RC5 CBC mode에서 암호화 및 복호화를 위해서 태그 시뮬레이터에서 암호화와 복호화에 사용하기 위한 라운드 키를 생성하기 위한 소요시간 측정결과를 (그림 13)에 나타내었다. 단위는 micro-sec이다.



(그림 13) 태그 시뮬레이터의 처리시간

제안한 모델의 처리 시간에 대한 이해를 돕기 위해서 펜티엄4-3.2Ghz 데스크 탑에서 동일한 처리를 위한 소요시간 측정결과를 (그림 14)에 나타내었다.



(그림 14) 데스크 탑에서의 처리시간

태그 시뮬레이터와 데스크 탑의 하드웨어적인 자원이 약 300배의 차이가 난다. 이를 고려할 때, 암호화와 복호화에 대한 처리 시간을 측정한 결과 성능의 차이는 약 600배의 차이를 보이고 있다.

결론으로 하드웨어적으로 태그 시뮬레이터 환경이 매우 열악한 점을 감안할 때 제안한 모델에서 대칭키 기반의 가볍고 빠른 암호 연산을 이용

해 인증과 위치추적 방지가 가능함을 보인 것으로서 본 논문에서 제안한 모델의 현실적인 적용 가능성을 보이고 있다.

추가적인 연구를 진행하여 처리 속도와 능동형 태그에서의 실험에 대한 비교 등이 수행된다면 더욱 좋은 실험결과를 보일 수 있을 것으로 사료된다.

참 고 문 헌

[1] S. Weis et al., "Security and Privacy Aspects of Low-cost Radio Frequency Identification Systems", Security and Pervasive Computing, LNCS 2802, pp. 201-212, 2003.

[2] L. Eschenauer and V. Gligor, "A Key-management Scheme for Distributed Sensor Networks", ACM CCS'02, Nov. pp. 41 47, 2002.

[3] H. Chan, A. Perrig and D. Song, "Random Key Pre-distribution Schemes For Sensor Network", IEEE Symposium on Security and Privcy, 2003.

[4] Tom Leighton and Silvio Micali, "Securet-Key Agreement without Public_ Key Cryptography", Advaces in Cryptology CRYPTO 1993, 1994.

[5] Duncan S. Wong and Agnes H. Chan., "Efficient and mutually authenticated key exchange for low power computing devices", In Advances in Cryptology ASIACRYPT' 2001.

[6] T. Leighton and S. Micali, "Secret-Key Agreement without Public-Key Cryptography", Advances in Cryptology CRYPTO 1993, pp. 456-479, 1993.

[7] 한국전산원, "전파식별(RFID) 보급 활성화를 위한 역기능 및 정보보호 대책 연구", 2004.

[8] 이용필, "RFID 사용확대에 따른 프라이버시 보호의 정책 및 기술적 대응 방안", 한국정보보호진흥원 전자거래보호 이슈리포트, 2000.

[9] 오수현, 곽진, "유비쿼터스 환경에 적합한 사용자 프라이버시 보호 기능을 제공하는 RFID 시스템", 한국통신학회논문지, pp. 1729-1738.

[10] 김진묵, 유황빈, "Random key Pre-Distribution protocol 기반의 안전한 RFID 시스템 설계", 전자공학회 논문지, 제42권, CI편, 제6호, 2005.

김 진 묵

1998년 배재대학교 컴퓨터과학과(이학사)

2000년 배재대학교 컴퓨터공학과(공학석사)

2006년 광운대학교 컴퓨터과학과(공학박사)

유 황 빈

1975년 인하대학교 전자공학과 (공학사)

1977년 연세대학교 대학원 (공학석사)

1989년 경희대학교 대학원 (공학박사)

1981년~현재 광운대학교 컴퓨터소프트웨어 교수