RFID 프라이버시 보호에서 병행성을 이용한 확장성 개선

신명숙* · 이 준*

Improving Scalability using Parallelism in RFID Privacy Protection

Myeong-Sook Shin* · Joon Lee*

요 약

RFID 시스템에서 프라이버시 침해 문제를 해결하기 위한 방안 중 백엔드 서버에서의 필수요건인 확장성을 단축하는 기법을 제안한다. 현재 RFID/USN이 큰 이슈가 되면서 RFID에 대한 각종 연구와 응용들이 활발히 진행 중에 있다. 반면에 RFID의 낮은 연산능력과 기억능력으로 개인의 프라이버시 보호 측면에서 여러 문제들을 유발시킨다. 기존 해시 체인 기법은 프라이버시를 침해하는 공격들에 대해서 전방 보안성, 기밀성, 불구분성 등을 모두 보장하는 안전한 기법이다. 그러나 백엔드 서버에서 태그를 식별하기 위한 계산량이 많다는 문제점이 있다. 따라서 본 논문에서는 백엔드 서버에서의 계산량을 감소하기 위해 키를 효율적으로 찾는 Hellman Method를 적용한다. Hellman Method 알고리즘은 선행계산과 탐색 두 단계로 진행되는 알고리즘이다. 본 논문에서는 해시 체인 기법에 Hellman Method를 적용한 후 병행성을 분석하고 분할적용하여 보안성과 키 검색을 비교하였다. 비교 결과는 기존의 프라이버시 보호를 위한 보안 요건을 모두 보장하면서 키 검색 비교는 기존 방식보다 계산 복잡도를 $O(m)$에서 $O(\frac{m^{2/3}}{w})$으로 단축하여 서버에서의 계산 시간을 단축하였다.

ABSTRACT

In this paper, we propose the scheme solving privacy infringement in RFID systems with improving the scalability of back-end server. With RFID/USN becoming important subject, many approaches have been proposed and applied. However, limits of RFID, low computation power and storage, make the protection of privacy difficult. The Hash Chain scheme has been known as one guaranteeing forward security, confidentiality and indistinguishability. In spite of that, it is a problem that requires much of computation to identify tags in Back-End server. In this paper, we introduce an efficient key search method, the Hellman Method, to reduce computing complexity in Back-End server. Hellman Method algorism progresses pre-computation and (re)search. In this paper, after applying Hellman Method to Hash chain theory, We compared preservation and key reference to analyze and apply to parallel With guaranteeing requistes of security for existing privacy protecting Comparing key reference reduced computation time of server to reduce computation complex from $O(m)$ to $O(\frac{m^{2/3}}{w})$than the existing form.

키워드

RFID, Privacy Protection, Hash Chain Scheme, Hellman Method, Parallelism

* 조선대학교 전자정보공과대학 컴퓨터공학과

접수일자 : 2006. 3. 24

Ⅰ. 서 론

RFID(Radio Frequency IDentification)[1]가 산업 전반에 걸쳐 다양하게 적용되기 시작하면서부터 프라이버시에 대한 중요성이 크게 부각되고 있다. RFID 시스템에서 프라이버시 침해 문제[2]를 해결하기 위해서는 최소한 3가지 보안 요건인 기밀성, 불구분성, 전방 보안성을 만족해야 한다. 또한 프라이버시 보호 기법을 적용하게 될 때, 백엔드 서버에서 보장하여야 하는 필수 요건은 확장성이다. 확장성[3]이란 처리해야 되는 전체 태그의 개수가 급격히 늘어난다 할지라도 적절한 시간 안에 식별 작업을 완수할 수 있어야 한다는 의미이다. 일반적으로 프라이버시 보호 기법을 설계하는 데 있어서, 안전성을 조금 더 높여주기 위해서는 백엔드 서버에서 처리해야 하는 계산량을 더욱 많이 늘려 주어야 한다. 백엔드 서버의 계산량이 어느 정도 이상 많아지게 되면 태그를 실시간으로 식별하는 것이 불가능해지기 때문에, 프라이버시 보호 기법의 연구에 있어서 백엔드 서버의 성능 측면에 대한 고려가 반드시 이루어져야만 한다.

기존 연구들 중 프라이버시 보호를 위한 필수 보안 요건 측면에서 살펴본다면 해시 체인 기법[4]이 가장 안전하다. 반면에 백엔드 서버에서는 태그를 식별하기 위한 계산량이 많다는 문제가 있다. 즉 확장성에 큰 문제가 있다고 할 수 있다. 결론적으로 지금까지의 연구에서는 위에서 말한 필수 보안 요건과 서버의 필수 요건인 확장성을 모두 만족시키는 기법이 제안되지 않는 상태이다.

본 논문에서는 프라이버시 보호를 위한 해시 체인 기법에 백엔드 서버에서의 계산량을 줄이기 위해서 Hellman Method[5]를 적용한 후 병행성[6]을 분석하여 키 검색 시간을 단축한 기법을 제안한다.

Ⅱ. RFID 구성 요소

RFID 시스템은 그림 1과 같이 태그 T_t(Tag), 리더 R(Reader) 그리고 백엔드 서버 B(Back-End Server)로 구성된다.

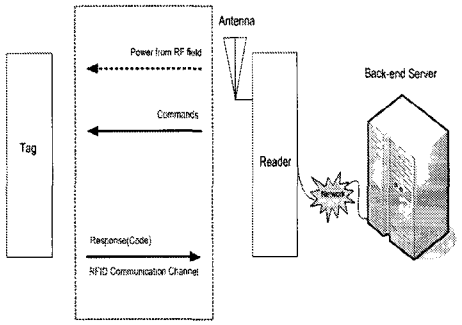

그림 1. RFID 시스템
Fig. 1. RFID System

2.1. RFID 시스템

RFID 시스템에서 태그와 리더 통신은 능동형 태그와 수동형 태그로 나누어질 수 있다. 능동형 태그는 태그 내 자체 배터리로 동작하는 형태이고, 수동형 태그는 리더에서 오는 반송파로 에너지를 얻어 동작하는 형태이다. 태그에서 전송된 데이터는 리더에게 전달되고 리더는 태그 정보를 확인하기 위해 백엔드 서버 시스템에서 태그 정보를 넘긴다. 일반적으로 배터리가 없는 수동형 태그는 능동형 태그에 비해 전송거리가 짧다는 단점이 있다.

백엔드 서버는 다수의 리더로부터 전송되어 오는 태그 관련 정보에 대한 처리를 해주는 시스템이다. 또한 태그와 관련된 정보를 데이터베이스화해서 관리하며 효율성을 위해서 여러 개의 서버로 분산 운영될 수 있다. 일반적으로 백엔드 서버가 보안 측면에서 신뢰할 수 있는 시스템이라고 간주한다. 백엔드 서버에서의 데이터베이스 관리 방식과 추가적인 계산 여부는 태그의 암호화 프로토콜에 따라서 많은 차이가 나게 된다.

2.2. RFID 인증 프로토콜

RFID 시스템에서 인증이란 태그와 리더의 정당성을 상호 입증해주는 절차를 의미하며, 태그와 리더는 서로를 인증하며 서로를 신뢰하는 경우에만 올바른 동작을 보장해야 하는데, 이는 접근제어(Access Control)와도 동일한 개념이다.

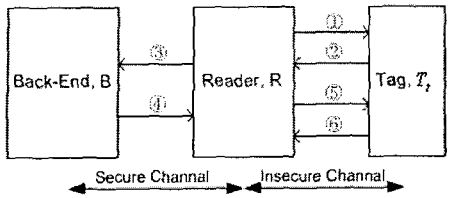

그림 2. RFID 인증 프로토콜 모델
Fig. 2. Model of RFID Authentication Protocol

그림 2와 같이 리더 R이 질의를 하면 태그 T_t는 현재의 자신을 인증시키기 위한 최소한의 정보를 리더 R에게 송신한다. 리더 R은 태그 T_t로부터 수신한 값을 그대로 안전한 채널을 이용하여 백엔드 서버 B에게 전송한다. 백엔드 서버 B는 태그를 인증하는 값을 찾아서 리더 R에게 전송해준다. 마지막으로 리더 R은 태그 T_t에게 태그가 원하는 정보를 보낸다. 태그 T_t 역시 보내온 값을 확인한 후 태그 T_t 자신의 정보를 리더 R에게 보낸다. 통상적으로 태그-리더 구간은 안전성이 보장되지 않아 도청이 가능한 구간이라 볼 수 있으며, 리더/서버 구간은 보안이 유지되는 안전한 구간이라 볼 수 있다.

Ⅲ. 관련연구

3.1. 해시 체인 기법

M. Ohkubo 등이 제안한 기법으로 그림 3과 같이 일방향 해시 함수를 사용하여 위치트래킹 공격에 안전하며, 전방 보안성이 보장되는 기법이다.

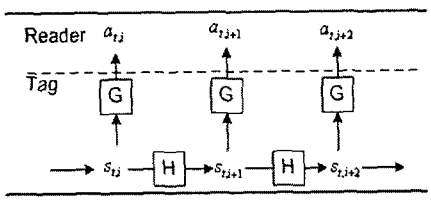

그림 3. 해시 체인 기법
Fig. 3 Hash Chain Protocol

B에는 태그 t에 대한 각각의 식별 정보 ID_t와 해시 시드값 $s_{t,1}$이 저장되며 태그에는 동일한 $s_{t,1}$값을 저장하고, 두 개의 해시함수 H와 G로 구현한다. R의 질의에 대해 태그는 $a_{t,i} = G(s_{t,i})$를 수행하여 R에게 응답하며 자신의 비밀 값인 $s_{t,i}$는 $H(s_{t,i})$를 통해 $s_{t,i+1}$로 갱신한다. R의 질의에 대해 T_t는 매번 다른 응답을 하므로 공격자는 T_t의 경로를 파악할 수 없게 된다.

그러나 이 기법에서 서버는 T_t의 ID_t와 초기 비밀 값인 $s_{t,1}$만을 가지고 있으므로 T_t로부터 온 $a_{t,i} = G(s_{t,i})$ 값에 해당하는 ID_t를 검색하기 위해서는 최악의 경우, 그림 4와 같이 B가 보유한 모든 $s_{t,i}$에 대해서 H와 G를 i번 수행해야 한다. B에서는 하나의 T_t를 식별하기 위한 계산에서, 모든 $1 \leq t \leq m$과 $1 \leq i \leq n$에 대해서 $a'_{t,i} = G(H^{i-1}(s_{t,1}))$를 계산하여야 하는데, 이를 계산 복잡도로 나타내면 $O(mn)$이 된다. 이것은 초당 2^{24}번의 해시를 하는 일반적인 해시함수의 성능 ($h_{speed} = 2^{24} times/sec.$)을 기준으로 계산한다[4]. 따라서 이 기법에서는 B에서 태그를 식별하기 위한 계산량이 많다는 것이다.

$$
\begin{array}{l}
s_{1,1} \rightarrow a_{1,1} \rightarrow \cdots \rightarrow a_{1,i} \quad \cdots \rightarrow \cdots \rightarrow a_{1,n} \\
\vdots \qquad\qquad\qquad\qquad \vdots \\
s_{t,1} \rightarrow a_{2,1} \rightarrow \cdots \rightarrow a_{t,i} \quad \cdots \rightarrow \cdots \rightarrow a_{t,n} \\
\vdots \qquad\qquad\qquad\qquad \vdots \\
\quad \cdots \quad a_{t,i} = G(H^{i-1}(s_{t,1})) \quad \cdots \\
\vdots \qquad\qquad\qquad\qquad \vdots \\
s_{m,1} \rightarrow a_{m,1} \rightarrow \cdots \qquad \cdots \rightarrow \cdots \rightarrow a_{m,n}
\end{array}
$$

그림 4. 백엔드 서버가 매번 계산해야 하는 해시 체인들
Fig. 4 Hash chains of Back-end server to compute each time

또한 T_t의 개수 m이 어느 정도 이상 커지면 적절한 시간에 T_t 식별이 불가능해지는 문제점이 있다.

3.2. Hellman Method 알고리즘

Hellman Method는 $K_0 \in \{0,1\}^k$인 임의의 선택키 K_0를 선택하고 선택된 키를 이용하여 평문을 암호문으로 만든다. 만들어진 암호문을 키로 사용하기 위해서 h

비트의 블록을 k 비트로 바꾸어 주는 변환함수 g 를 적용하여 키를 생성한다. 이러한 일련의 과정을 n 동안 반복 수행한다. 이 때 g 함수는 암호문을 치환하고 64비트를 56비트로 줄여주는 함수로서 일 방향 함수이고 n을 depth라 하며 키 K_0를 n depth 반복된 것을 체인이라 한다.

선택된 평문 P를 키 K_0를 이용하여 DES로 암호화하고 h 비트의 블록 암호문을 k 비트로 변환하는 함수를 f 함수로 표기하면 f 함수는 식 (1)과 같이 정의된다.

$$g : \{0,1\}^h -> \{0,1\}^k$$
$$f(K_0) = g(E_{K_0}(P)) \quad (1)$$

키 K_0와 f 함수의 관계를 식으로 나타내면 식 (2)와 같다.

$$K_i = f(K_{i-1}) = f^i(K_0), i = 1,2...,n \quad (2)$$

여기서 K_0를 SP(Starting Point), K_n을 EP(Ending Point)라 한다. m개의 키를 갖는 경우 $(mn)^{2/3}$의 메모리와 $(mn)^{2/3}$의 연산으로 키를 찾을 수 있다. 즉 전수조사보다는 시간이 적게 걸리고 미리 계산된 테이블을 갖고 있을 경우보다 적은 메모리를 필요로 한다. Hellman Method는 크게 선행계산 과정과 키 검색 과정으로 나뉜다.

3.2.1 선행계산

그림 5와 같이 r 개의 다른 g함수에 따라 m개의 SP를 depth n 동안 반복하여 EP를 생성하고 생성된 EP에 대하여 정렬한 후 (SP,EP)를 저장하여 g 함수에 대한 테이블을 만든다. 또한 동일한 m 개의 SP에 다른 g 함수에 따른 테이블을 만든다. 이상적인 경우 $m = 2^{32}$이고 $n = 2^{32}$일 때 2^{64}개를 생성한다. 결국 각각의 g 함수에 따라 (SP,EP)의 테이블을 생성함으로써 기억공간의 부담을 줄인다.

$$SP_1 = K_{1,0} \xrightarrow{f} K_{1,1} \xrightarrow{f} K_{1,2} \xrightarrow{f} \cdots\cdots \xrightarrow{f} K_{1,n} = EP_1$$
$$SP_2 = K_{2,0} \xrightarrow{f} K_{2,1} \xrightarrow{f} K_{2,2} \xrightarrow{f} \cdots\cdots \xrightarrow{f} K_{2,n} = EP_2$$
$$\vdots \qquad\qquad \vdots \qquad\qquad \vdots$$
$$SP_m = K_{m,0} \xrightarrow{f} K_{m,1} \xrightarrow{f} K_{m,2} \xrightarrow{f} \cdots \xrightarrow{f} K_{m,n} = EP_n$$

그림 5. (SP,EP) 쌍 생성
Fig. 5 Generation of (SP,EP) Pair

선행계산 알고리즘은 그림 6과 같다.

```
for s = 1 to r
  for t = 1 to m
    SP_t^(s) 생성
    K_0 = SP_t^(s)
    for i = 1 to n
      K_i = f_K^i(E_{K_{i-1}}(P))
    next i
    EP_t^(s) = K_n
  next t
next s
```

그림 6. 선행계산 알고리즘
Fig. 6 Precomputation Algorithm

해시 체인 기법 적용에 있어서는 $a_{t,i}$가 B에 전달되어지면, $a_{t,i}$에 대해서 H와 G를 번갈아 적용해가면서 새로운 체인들의 끝 값과 비교를 하면서 일치하는 값을 찾게 되고, 찾으면 $a_{t,i}$가 위치하고 있는 SP의 첫 값부터 다시 계산해가서 $a_{t,i}$를 만들어내게 된 t값과 i 값을 알아낼 수 있게 된다. 해시 체인 기법에서 f 함수를 $f = (t,i) -> a_{t,i} = G(H^{i-1}(s_{t,1}))$로 정의할 수 있다.

테이블 선행계산 방법은 모든 키에 대해 암호문을 저장하므로 공간 복잡도가 상당히 크지만 Hellman Method에서는 EP를 계산하기 위한 과정에서 생성된 키를 저장하지 않고 (SP,EP) 쌍만 저장하므로 공간 복잡도가 이보다 크지 않다.

3.2.2 키 검색

임의의 암호문을 획득하였을 경우 선행계산 과정을 통해 생성한 (*SP,EP*) 쌍을 이용하여 획득한 암호문을 암호화하기 위해 사용된 키를 알아내는 과정이 키 검색 과정이다. 이 과정은 암호문 $s_{t,1}$를 찾는 단계로서 먼저, 변환함수 f 함수를 적용하여 $a'_{t,i} = G(H^{i-1}(s_{t,i}))$를 계산하고 $t \in \{1,..,m\}$인 $a_{t,i}? = a'_{t,i}$을 비교하여 같으면 f 함수를 적용하여 $n-1$번 계산된 $s_{t,1}$를 찾는다. $a_{t,i}? = a'_{t,i}$가 같지 않으면 이러한 과정을 $n-1$ 동안 반복한다.

키를 알아내는 과정은 (*SP,EP*)의 쌍이 EP의 값에 대해 정렬되어 있으므로 바이너리 탐색을 통해 탐색을 수행하는 것은 그림 7과 같다.

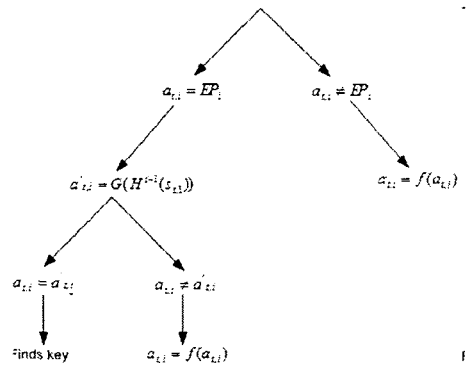

그림 7. 키 검색 알고리즘
Fig. 7 Key Search Algorithm

결국 본 논문에서 적용한 해시 체인 기법에서는 T_t에서 생성된 값 $a_{t,i}$ 가무선 통신을 통해 리더에게 전달되고, 리더는 T_t로부터 받은 $a_{t,i}$ 값을 안전한 채널로 B 시스템에게 보낸다. B는 모든 $s_{t,1}$와 모든 $t(1 \leq t \leq m)$와 $i(1 \leq i \leq n)$에 대하여 값을 가지고 식 (3)번과 (4)번 과정을 수행하여 EP를 찾아낸 다음 전송 받은 $a_{t,i}$와 일치하는 $s_{t,1}$ 값을 찾아낸다.

$$a'_{t,i} = G(H^{i-1}(s_{t,i})) \tag{3}$$

$$a_{t,i}? = a'_{t,i} \tag{4}$$

따라서 Hellman Method에서 m이 SP의 수행횟수이고 n은 EP를 구하기 위한 반복횟수일 때,

$m = n = N^{\frac{1}{3}}$일 때, 키 탐색 계산 복잡도는 $O(N^{\frac{2}{3}})$로 감소된다.

Ⅳ. 제안 기법

기존의 해시 체인 기법은 기밀성과 불구분성 그리고 전방 보안성이 보장되지만 벡엔드 서버의 확장성에서는 $O(mn)$의 계산적 복잡도를 가지게 된다. 제안 기법에서는 먼저 Hellman Method에서 병행성을 분석하고 작업을 분할한 다음 해시 체인 기법에 적용한다. 이는 각각의 작업을 독립적으로 수행할 수 있도록 동일한 크기인 w개의 노드로 분할하여 동시에 수행한다면 이상적인 경우 작업 완료 시간을 $\frac{1}{w}$로 감소함으로써 백엔드 서버에서의 계산 시간을 단축한다.

4.1. 병행성 분석 및 적용

병행성 적용을 위한 Hellman Method 알고리즘은 전체 수행 시간 중 (*SP,EP*) 쌍을 생성하기 위해 소요되는 시간이 가장 많은 부분을 차지한다. 따라서 (*SP,EP*) 쌍을 생성하기 위해서 소요되는 시간을 단축함으로써 작업 시간을 크게 단축할 수 있다.

그림 8은 Hellman Method의 선행계산 과정에서 *SP*로부터 *EP*를 독립적으로 계산하는 과정을 나타냈다. 즉, 서로 다른 *SP*로부터 *EP*를 계산하는 과정에서 서로 간섭이나 종속성이 전혀 없다. 따라서 임의의 *SP*를 선택하는 단계를 각 노드에서 동시에 수행할 수 있다. 즉 병렬 처리로 선행계산과 키 검색 과정을 적절히 분할하여 적용 가능하다.

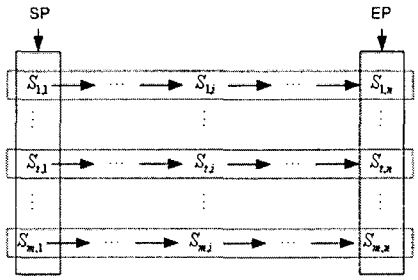

그림 8. 각각의 SP로부터 EP를 독립적 계산
Fig. 8 Separation Computation of EP from each SP

그림 9는 임의의 *SP*를 선택하고 (*SP*,*EP*) 쌍을 계산하는 과정을 위한 작업을 분할하는 방법을 나타낸 것이다. 이 때 *SP*의 개수가 m개 일 경우 w개의 노드가 이를 분할하여 수행하게 되므로 각 노드에 동일하게 작업을 부여할 경우 각 노드에서 선택할 *SP*의 개수는 $\frac{m}{w}$이 된다.

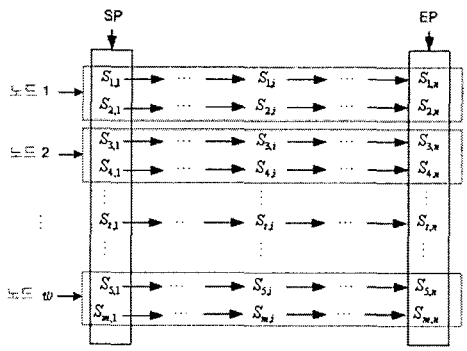

그림 9. Hellman Method의 작업분할 방법
Fig. 9 Operation Partition Method of Hellman Method

병행성을 이용하여 다중 노드에서 독립적으로 Hellman Method를 수행하는 과정은 먼저 $\frac{m}{w}$개의 임의의 *SP*를 선택하여 *EP*를 계산한 후 *EP*에 대해 정렬한다. 그리고 나서 키를 탐색하는 과정을 노드별로 수행한다. 따라서 위의 과정은 병행성을 적용함으로써 독립적으로 수행한다.

4.2. 백엔드 서버에서의 계산 복잡도 분석

본 논문에서 제안하는 기법으로 백엔드 서버에서는 하나의 태그를 식별하는 데 필요한 계산량을 분석하여 확장성의 향상 정도를 설명한다.

독립적으로 수행할 수 있는 작업을 분할하여 동시에 수행할 수 있다면 전체 *SP*의 개수가 m개 일 경우 w 개의 노드로 분할하여 각 노드에 동일하게 작업을 부여한다면 제안 기법이 각 노드에서 선택할 SP의 개수는 $\frac{m}{w}$이 된다. 또한 하나의 태그를 식별하기 위한 키 검색을 하는 계산 복잡도를 분석하면 기존의 해시 체인 기법은 $O(mn)$이 되고, 제안 기법은 $O(\frac{m^{2/3}n^{2/3}}{w})$이 된다. 예를 들어서, 태그의 개수가 2^{20}이고, 해시 체인의 최대 길이가 2^{10}, 1초당 수행할 수 있는 해시 함수의 횟수를 $h_{speed}=2^{24}$ *times/sec.*라고 가정했을 때[4], 하나의 태그를 식별하기 위해서 백엔드 서버가 계산하는데 걸리는 시간은 2^6초 (=64초)이고, 제안 기법에서는 시스템이 단일 컴퓨터에 의존한다고 가정하고 $w=4$일 때 이상적일 경우 2^{-6}초 ($=0.015625$초)이다. 노드(w) 수가 증가하면 시간은 더 감소할 수 있다.

<표 1>은 프라이버시 보호를 위한 필수 요건을 모두 만족하는 기법에 대해서 백엔드 서버의 해시 체인 기법과 제안 기법을 계산 복잡도로 표현되어 있는 값들을 실제적인 값을 통해서 비교한다. 노드(w) 수는 4개로 제한하였고 해시 함수의 계산 측면에서 태그의 개수(m)는 무한정으로 많아지는데 n은 어느 정도의 범위, 즉 1000, 10000, 100000 ... 등 상수의 범위를 가진 값이기 때문에 상수항으로 보고 분석한다.

표 1. 백엔드 서버에서 해시 체인 기법과 제안 기법의 보안 요건 및 키 검색 비교
Table. 1 Comparison of Key Search and Security Requirement of Hash Chain Scheme and Proposal Scheme of Back-end server

		해시체인기법	제안 기법
프라이버시 보호 보안 요건	기밀성	O	O
	불구분성	O	O
	전방 보안성	O	O
키 검색 계산 복잡도		$O(m)$	$O(\frac{m^{2/3}}{w})$
태그 판별에 걸리는 시간		64 초	0.015625 초

m : 태그 개수, w : 노드 수($w=4$)

Ⅳ. 결 론

RFID 시스템에서 프라이버시 침해 문제를 해결하기 위한 방안 중 백엔드 서버에서의 필수 요건인 확장성을 단축하는 새로운 기법을 제안하였다.

기존의 기법들 가운데 프라이버시 보호를 위한 필수 보안 요건을 모두 보장하는 안전한 기법은 해시 체인 기법이다. 그러나 이 기법은 백엔드 서버의 확장성 측면에서 문제가 있기 때문에 실제 RFID 시스템의 프라이버시 보호 기법으로 활용되기에는 어려움이 있다. 반면에 이 논문에 적용된 병행성은 Hellman Method에서 키를 알아내는 과정 중 서로 다른 SP에 대해서는 전혀 종속성이 존

재하지 않는 점을 이용하여 동시에 수행할 수 있으며, 각 노드에서 선택된 SP를 이용하여 (SP,EP) 쌍을 계산하는 과정과 키 검색 과정을 독립적으로 수행한 결과 키 검색 시간을 단축하였다.

따라서 본 논문에서 제안하는 기법은 백엔드 서버에서 태그의 *ID*를 식별하는데 있어서 병행성을 이용한 결과 보안 요건은 모두 만족하면서 키 검색 시간은 기존 기법보다 계산 복잡도를 $O(m)$에서 $O(\frac{m^{2/3}}{w})$으로 단축하였다. 즉 하나의 태그를 식별하기 위해서 백엔드 서버가 계산하는데 걸리는 키 탐색 시간은 이상적일 경우에 2^6초에서 2^{-6}초($w=4$)로 감소하여 수행 시간을 단축하였다. 이러한 확장성 단축으로 안전하고 효율적인 RFID 인증 프로토콜이 요구하는 시스템 사양을 쉽게 적용할 수 있다는 장점이 있다. 또한 제안 기법은 안전성과 효율성 뿐만 아니라 분산 환경을 고려하여 유비쿼터스 환경 실현을 위해 다양하게 활용될 수 있을 것으로 기대된다.

참고문헌

[1] Association for Automation Identification and Data Capture Technologies (AIM) (2002), Shrouds of Time: The History of RFID. http://www.aimglobal.org, Last accessed August 22, 2004.

[2] 이승구, 여상수, 조정직, 김성권, "RFID 시스템에서 안전하고 효율적인 프라이버시 보호 기법",한국컴퓨터종합학술대회, Vol 32, No 1(A), 2005년.

[3] Gildas Avoine and Philippe Oechslin. "A scalable and provably secure hash based RFID protocol". In IEEE International Workshop on Pervasive Computing and Communication Security - PerSec 2005, Kauai Island, Hawaii, USA, March 2005. IEEE, IEEE Computer Society Press.

[4] M. Ohkubo, K. Suzuki, and S. Kinoshita. "Cryptographic approach to "privacy-friendly" tags". In RFID Privacy Workshop, MIT, USA, 2003.

[5] M. Hellman. "A cryptanalytic time-memory trade off". IEEE Transactions on Information Theory, IT-26(4): 401:406, 1980.

[6] 김이남, "병렬처리를 통한 AES 알고리즘에 관한 연구", 석사학위논문, Dec. 2002.

저자소개

신 명 숙(Myeong-Sook Shin)

1992년 광주대학교 전자계산학과 (공학사)
1996년 광주대학교 대학원 컴퓨터학과(공학석사)
2005년 조선대학교 대학원 컴퓨터 공학과(박사과정 수료)
2006년 - 현재 조선대학교 컴퓨터공학과 시간강사
※관심분야 : 시스템소프트웨어, 유비쿼터스컴퓨팅, 정보보호 등

이 준(Joon Lee)

1979년 조선대학교 전자공학과 (공학사)
1981년 조선대학교 대학원 전자공학과 (공학석사)
1997년 숭실대학교 대학원 전자계산학과(공학박사)
1982년 - 현재 조선대학교 전자정보공과대학 컴퓨터공학부 교수(현)
※관심분야 : 운영체제, 정보보호, 유비쿼터스컴퓨팅 등