그리드 환경에서 SP분할 알고리즘을 이용한 확장성 있는 RFID 태그 판별

신명숙* · 안성수** · 이준***

RFID Tag Identification with Scalability Using SP-Division Algorithm on the Grid Environment

Myeong-Sook Shin* · Seong-Soo Ahn** · Joon Lee***

요 약

최근 RFID 시스템의 채택이 다양한 분야에서 빠르게 진행되고 있다. 그러나 RFID 시스템의 대중화를 위해서는 RFID 태그의 정보를 무단으로 획득함으로써 발생할 수 있는 프라이버시 침해 문제를 해결해야 한다. 이 문제를 해결하기 위해서 기존 연구들 중에서 가장 안전한 M. Ohkubo 등의 Hash-Chain 기법이 있다. 그러나 이 기법은 태그를 판별할 때 엄청난 태그 수의 증가로 인해 막대한 계산 능력을 요구하는 문제점이 있다. 따라서 본 논문에서는 프라이버시 보호 기법의 필수 보안 요건 3가지를 모두 만족하면서 태그 판별 시간을 감소할 수 있는 방법을 제안한다. 먼저 이질적인 시스템으로 구성되는 그리드 환경의 특성으로 인해 최적화된 성능을 얻기 위하여 Hash-Chain 계산 테이블을 생성하는 프로그램을 작성한 후 각 노드들의 성능 측정한다. 그 측정 결과를 이용하여 SP들을 분할하는 SP분할 알고리즘을 제안한다. 또한, 구현 결과 Hash-Chain 길이 1000, 노드 수 4로 고정된 상태에서 측정한 결과를 단일 노드와 균등분할, SP분할을 비교하면 SP들의 총수가 1000개 일 때 40%, 49%, 2000개 일 때 42%, 51%, 3000개 일 때 39%, 49%, 4000개 일 때 46%, 55%가 향상되었다.

ABSTRACT

Recently RFID system has been adopted in various fields rapidly. However, we ought to solve the problem of privacy invasion that can be occurred by obtaining information of RFID Tag without any permission for popularization of RFID system To solve the problems, it is Ohkubo et al.'s Hash-Chain Scheme which is the safest method. However, this method has a problem that requesting lots of computing process because of increasing numbers of Tag. Therefore, We suggest the way(process) satisfied with all necessary security of Privacy Protection Shreme and decreased in Tag Identification Time in this paper. First, We'll suggest the SP-Division Algorithm seperating SPs using the Performance Measurement consequence of each node after framing the program to create Hash-Chain Calculated table to get optimized performance because of character of the grid environment comprised of heterogeneous system. If we compare consequence fixed the number of nodes to 4 with a single node, equal partition, and SP partition, when the total number of SPs is 1000, 40%, 49%, when the total number of SPs is 2000, 42%, 51%, when the total number of SPs is 3000, 39%, 49%, and when the total number of SPs is 4000, 46%, 56% is improved.

키워드

RFID, Computational Grid, MPICH-G2, Privacy Protection, Hash-Chain Scheme

* 조선대학교 전자정보공과대학 컴퓨터공학과
** 동신대학교 전기공학과
*** 조선대학교 전자정보공과대학 (교신저자)

접수일자 : 2009. 04. 09
심사완료일자 : 2009. 07. 06

Ⅰ. 서 론

RFID가 산업 전반에 걸쳐 다양하게 적용되기 시작하면서부터 프라이버시에 대한 중요성이 크게 증가되고 있다. 현재 사용되고 있는 RFID[1]는 RFID 시스템이 가지고 있는 특성으로 인하여 사용자 프라이버시 문제[2]를 발생시킨다. 이러한 RFID 시스템에서 프라이버시 침해 문제를 해결하기 위해서는 보안 요건들, 즉 기밀성, 불구분성, 전방 보안성을 모두 만족해야 한다. 또한 프라이버시 보호 기법을 적용하게 될 때, 백엔드 서버에서 보장해야 하는 필수 요건은 확장성이다. 확장성이란 처리해야 되는 전체 태그의 개수가 급격히 늘어난다 해도 적절한 시간 안에 태그 판별 작업을 완수할 수 있어야 한다는 의미이다. 일반적으로 프라이버시 보호 기법을 설계하는데 있어서, 안전성을 조금 더 높여주기 위해서는 백엔드 서버에서 처리해야 하는 계산량을 더욱 많이 늘려주어야 한다. 그러나 백엔드 서버의 계산량이 어느 정도 이상 많아지게 되면 태그를 실시간으로 판별하는 것이 불가능해지기 때문에, 백엔드 서버의 성능 측면에 대한 고려가 반드시 이루어져야만 한다. 따라서 컴퓨터 및 네트워크 성능이 향상됨에 따라 지역적으로 분산되어있는 고성능의 컴퓨터자원의 공유를 통해서, 대규모의 데이터를 계산 처리할 수 있는 방법에 초점을 모으고 있다.

기존 연구들 중에서 프라이버시 보호를 위해 저가의 태그를 이용한 RFID 시스템 환경에서 가장 안전한 기법으로 M. Ohkubo 등의 Hash-Chain 기법[3]을 이용하였다. 그러나 이 기법은 백엔드 서버에서 태그 ID를 판별하기 위하여 모든 태그에 대한 정보를 가지고 순차적으로 판별 과정을 수행한다. 실제로 태그 수의 증가로 인해 막대한 계산 능력을 요구하는 문제점을 가지고 있다. 여기에서 m*n Hash-Chain 계산 테이블에서 해시 시드 값 $s_{1,1}, s_{2,1}, \cdots s_{m,1}$을 SP(Startpoints)라고 하고 n번 연산을 마친 마지막 값, $s_{1,n}, s_{2,n}, \cdots s_{m,n}$을 EP(Endpoints)라고 가정한다.

따라서 기존 계산 작업의 한계를 극복하기 위한 대안으로 지리적으로 분산되어 있는 고성능 컴퓨팅 자원을 네트워크로 상호 연동하여 조직과 지역에 관계없이 가상 집합체들의 자원을 공유하여 대량의 태그 처리를 수행할 수 있는 계산 그리드(Computational Grid)[4]로의 이식 작업이 필요하다.

본 논문에서는 프라이버시 보호를 유지하면서 이질적인 시스템으로 구성되는 그리드 환경에서 최적화된 성능을 얻기 위하여 Hash-Chain 계산 테이블을 생성하는 프로그램을 작성하여 각 노드의 성능을 측정한 후 그 결과를 이용하여 SP(Startpoints)를 분할하는 SP분할 알고리즘을 제안하여 구현한다.

Ⅱ. 관련 연구

본 장에서는 RFID 프라이버시 보호를 위한 계산 그리드 환경의 태그판별처리 모델에 적합한 연구들을 소개한다.

2.1 Hash-Chain 기법

M. Ohkubo 등이 제안한 기법[3]으로 그림 1과 같이 일방향 해시 함수를 사용하여 안전한 프라이버시 보호가 보장되는 기법이다.

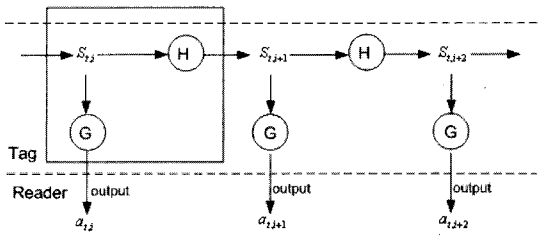

그림 1. Hash-Chain 기법
Fig. 1. Hash-Chain Scheme

백엔드 시스템에는 ID_i와 해시 시드 값 $s_{i,1}$이 저장되며 태그에도 동일한 $s_{i,1}$값을 저장하고, 두 개의 해시함수 H와 G로 구현한다. 리더의 질의에 대해 태그는 $a_{i,i} = G(s_{i,i})$를 수행하여 리더에게 응답하며 자신의 시드 값인 $s_{i,i}$는 $H(s_{i,i})$를 통해 $s_{i,i+1}$로 갱신한다. 그러나 이 기법은 서버에서 태그를 판별하기 위한 계산량이 많다는 문제점이 있다.

2.2 그리드 컴퓨팅

그리드 컴퓨팅[5]은 지리학적으로 분산되어 있는 고성능 컴퓨팅 자원을 네트워크로 연동하여 조직과 지역에 관계없이 가용할 수 있는 컴퓨팅 환경을 말한다. 그리드 시스템은 계산 그리드와 데이터 그리드, 엑세스 그리

드로 나눌 수 있다. 계산 그리드는 분산되어 있는 고성능 컴퓨팅 자원을 연결하여 실행함으로써 지금까지 불가능하였던 연구를 수행할 수 있는 환경을 제공하고 High-Throughput 계산 환경을 구축하기 위한 고성능 컴퓨팅 자원을 통합하는 기술이다. 데이터 그리드는 한 곳에 집중된 대량의 데이터를 효율적으로 공유하고, 여러 곳에 분산되어 있는 대량의 데이터 및 데이터베이스에 실시간으로 접근하기 위하여 단계적이고 체계적으로 데이터를 접근 할 수 있게 해주는 기술이며, 액세스 그리드는 동일 분야 연구자들의 공동 연구나 정책 결정을 위한 다양한 분야 연구자들이 원격지에서 접근하여 의견을 교환하고 협력할 수 있는 고성능 협업 환경을 구축하는 기술이다.

2.3 Globus Toolkit

Globus는 지리적으로 분산된 이종적인 컴퓨팅 자원들을 하나의 가상 컴퓨터처럼 사용 가능할 수 있도록 하는 소프트웨어 기반 구조를 제공한다.

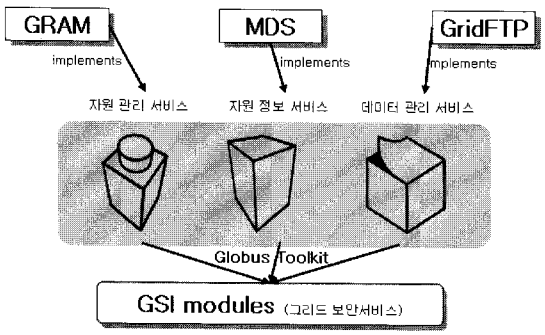

그림 2. Globus 구성 요소
Fig. 2. Globus Configuration Element

Globus 구성 요소는 그림 2와 같이 플랫폼에 직결된 GRAM, MDS, GridFTP, GSI 등 핵심 서비스만 제공하고 응용 프로그램이 자신의 목적에 필요한 서비스를 선택/조합함으로써 다양한 형태의 응용프로그램 및 패러다임 지원한다. 또한 지리적으로 분산된 컴퓨팅 자원들을 하나의 가상 컴퓨터처럼 사용할 수 있도록 소프트웨어 기반 구조를 제공한다.

2.4 MPICH-G2

MPICH-G는 MPI에 Globus가 제공하는 서비스를 가미함으로서 그리드 컴퓨팅에서의 병렬 프로그래밍 환경의 기초를 구성하였으나, 성능이 기존에 사용하던 벤더들의 MPI에 비해 현저하게 떨어지는 단점이 있어 널리 사용되기에는 큰 부족함이 있었다. 그 후 성능의 향상에 초점을 맞추어 지속적인 연구를 거듭하여 MPICH-G2 (Grid-enable MPI Chameleon 2)를 개발하였으며 MPICH-G2는 성능면에서 일반 벤더들의 MPI에 뒤떨어지지 않는다. MPICH-G2는 MPICH-G와 마찬가지로 Globus가 제공하는 많은 서비스를 이용하고 있지만 MPICH-G에서 사용되었던 Nexus를 제거함으로써 많은 성능을 향상 시켰다. Nexus는 여러 개의 프로토콜을 지원하고 자동적인 Data의 변환을 지원하는 등 여러 가지 매력적인 기능으로 오랫동안 MPICH-G의 통신 기반 구조로 사용되었지만 그 외 여러 가지 향상시켜야하는 기능들에 대한 문제로 제거 되었다. MPICH-G2는 모든 통신을 직접적으로 다루도록 재 구현 하였고 이를 통해 Nexus를 사용했을 때보다 큰 성능의 향상을 가져 왔다[6].

Ⅲ. 제안한 방법

기존의 Hash-Chain 기법은 안전한 보안성이 보장되지만 분산 환경에서 엄청난 태그 수의 증가로 인해 막대한 계산 능력을 요구하는 문제점이 있다. 이러한 문제점을 해결하기 위해서 Hash-Chain 기법의 병행성을 분석하여 그리드 환경으로 이식하고, 각 노드의 성능 측정 결과를 이용하여 SP를 분할하는 SP분할 알고리즘을 제안한다.

3.1 그리드 환경으로의 이식

RFID 프라이버시 보호를 위해 적용한 Hash-Chain 기법은 그림 3과 같이 하나의 태그를 판별하기 위한 계산에서, 백엔드 시스템에서는 모든 $1 \leq t \leq m$와 $1 \leq i \leq n$에 대해서 $a'_{t,i} = G(H^{i-1}(s_{t,1}))$를 계산한다.

Hash-Chain 기법에서 SP로부터 EP를 계산하는 과정은 그림 3과 같이 서로 다른 SP에 대해 독립적이다. 즉, 서로 다른 SP로부터 EP를 계산하는 과정에서 서로 간섭이나 종속성이 전혀 없으므로 이 과정은 동시에 수행될 수 있다. 또한 태그 판별 과정도 서로 종속성이 없이 독립적이다.

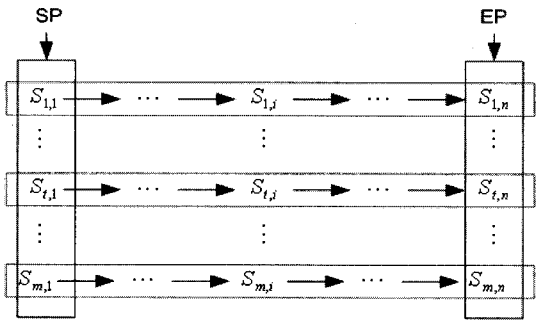

그림 3. 서로 다른 SP에 대해 독립적
Fig. 3. Independence from mutually different SP

본 논문에서는 태그 판별을 위해 막대한 계산 능력을 요구하기 때문에 계산 그리드를 활용하기에 적합하다. 계산 그리드를 이용하여 문제를 해결하기 위해서 소요되는 시간 단축을 위해서 제기된 문제의 병행성을 충분히 분석하여 동시에 수행될 수 있는 작업으로 분할할 수 있어야 한다. 계산 그리드를 이용하여 문제를 해결하기 위해서는 제기된 문제의 병행성을 분석하여 k개의 노드로 SP를 균등하게 분할한다. 이를 식으로 나타내면 식 (1)과 같다.

$$SP[i] = \frac{m}{k} \tag{1}$$

여기서 SP[i]는 분할된 SP들의 개수이고, m은 전체 태그 개수이다. 또한 k는 노드 수이며 i는 {1, 2, 3⋯k}이다. 동시에 수행될 수 있는 노드 수가 많을수록 그리드의 활용도는 높아진다. 그림 4는 계산 그리드 환경에서 k개의 노드로 SP를 균등하게 분할하는 방법을 나타낸 것이다.

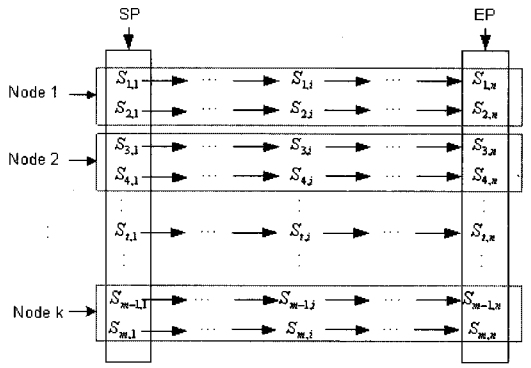

그림 4. 동일한 크기에 따른 균등분할 방법
Fig. 4. Even Division Method with Same Size

3.2 SP분할 알고리즘

기본적으로 k 개의 노드가 있을 경우, 각 노드에 SP들을 균등하게 분할하여 적용한 방법은 이질적인 시스템으로 구성되는 그리드 환경에서는 최적화된 성능을 얻을 수 없었다. 이를 해결하기 위한 방법으로 노드의 성능에 따라 SP들을 다양하게 분할하는 SP분할 알고리즘을 제안한다. SP분할 알고리즘에서 사용되는 변수와 관련된 식들은 다음과 같다.

표 1. SP 분할 알고리즘에서 사용되는 변수 정의
Table 1. Definition of Parameters used in SP-Division Algorithm

Parameter	Content
m	Total Number of SPs
k	Total Number of Nodes
i	{1, 2, 3, ⋯, k}
node[i]	Used Number of Nodes
perf[i]	Inverse Number of Performance Measurement Result of node[i]

식 (2)에서 각 노드에 대한 성능 지수 perf[i]는 해시 계산 시간의 역수이다.

$$perf[i] = \frac{1}{node[i]} \tag{2}$$

각 노드들의 성능을 측정하여 SP들을 분할하는 SP분할 비율은 식 (3)과 같다.

$$SP\ 분할비율 = \frac{perf[i]}{\sum_{i=1}^{k} perf[i]} \tag{3}$$

SP 분할비율에 따른 SP들의 분할은 식 (4)와 같다.

$$SP[i] = ROUND\left(\frac{m \cdot perf[i]}{\sum_{i=1}^{k} perf[i]}\right) \tag{4}$$

3.3 태그판별처리 모델

그리드 환경의 태그판별처리 모델은 그림 5와 같이 역할에 따라 Master와 Slave로 구성된다. Salve는 태그를 판별하는 역할을 한다. Master는 Slave에게 수행할 SP를 균등하게 할당하여 Slave로 전송하고 Slave로부터 생성된 결과를 수집하는 역할을 한다. 즉 Master는 Slave를 관리하는 역할을 한다.

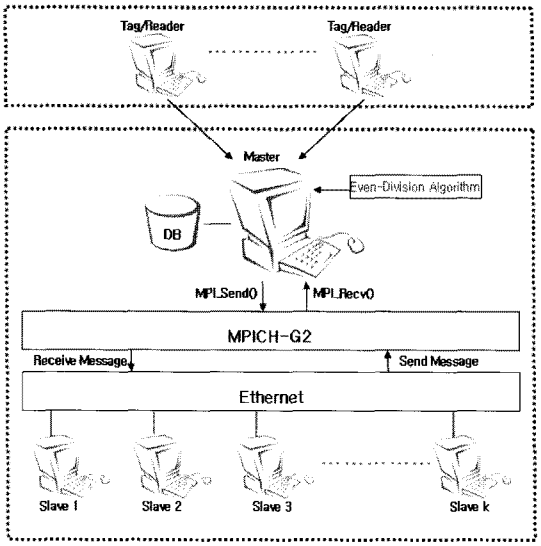

그림 5. 그리드 환경의 태그판별처리 모델
Fig. 5. Tag Identification Process Model of Grid Environment

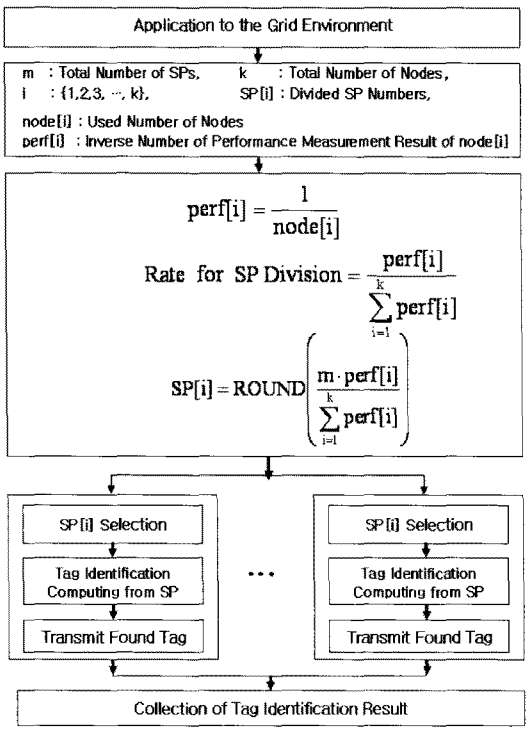

그림 6. SP 분할 알고리즘을 이용한 흐름도
Fig. 6. Flowchart using SP-Division Algorithm

또한 각각의 시스템의 운영체제는 RadHat Linux, 미들웨어는 Globus Toolkit과 MPICH-G2를 탑재하여 사용한다. Master와 Slave 사이는 LAN을 통해 연결되며 메시지 교환을 위해서는 MPI_Recv()와 MPI_Send() 함수를 이용한다.

그리드 환경에서 SP 분할 알고리즘을 바탕으로 전체적인 흐름은 그림 6과 같다. 기존 방법과 마찬가지로 태그/리더-그리드 간의 통신은 무선 전파를 이용한 통신이므로 도청의 가능성이 있는 것으로 간주하였으며, 그리드 환경의 마스터/슬레이브 간의 통신은 안전하다고 가정한다.

위에서와 같이 백엔드 서버에서의 태그판별시간을 단축하기 위해서는 작업을 균등하게 분할하여 그리드 환경으로 이식한다. 또한 각 노드의 성능 측정 결과를 이용하여 SP들을 분할하는 SP분할 알고리즘을 이용하여 각 노드별로 동시에 수행함으로써 대량의 SP를 처리할 때 태그판별처리 시간을 단축할 수 있는 모델을 제공한다.

Ⅳ. 실험 및 분석

4.1 실험 환경

성능평가를 위한 실험 환경은 표2와 같이 구성된다.

표 2. 구현을 위한 하드웨어 및 소프트웨어 환경
Table 2. Environment of Software and Hardware for Implement

항 목		내 용
하드웨어	Master	Intel Xeon 5130 2.0G, dual core 4.0GB
	Slave1	Pentium4 2.4G 512M
	Slave2	Pentium4 3.0G 512M
	Slave3	Pentium4 2.8G 512M
	Slave4	Intel Xeon 5130 2.0G, dual core 4.0GB
소프트웨어	운영체제	RadHat Linux 9.0, 커널버전 : 2.4.20-8)
	미들웨어	Globus Toolkit 2.2.2
	MPI	MPICH-G2
	언 어	C

여기서 제안 모델은 Master와 Slave로 구성된다. Master 역할을 하는 기계는 Intel Xeon 5130 2.0G, dual

core 4.0GB 기계를 사용한다. Master와 Slave는 LAN을 통해 연결되며 Hash-Chain 연산을 하여 태그를 판별하는데 이용한다. 그리고 Hash-Chain 연산에 사용한 일방향 해시 함수는 128bit의 md4, md5를 이용하였으며 검색 방법은 순차 검색을 이용하였다.

그리드 환경에서 메시지 전달은 Master와 Slave 사이의 통신을 위해서 필요하다. 본 논문에서는 그림 7과 같이 Master와 Slave 사이의 통신을 위해 MPICH-G2를 이용하였으며 MPI_Recv()와 MPI_Send() 함수를 이용하여 메시지 교환을 한다.

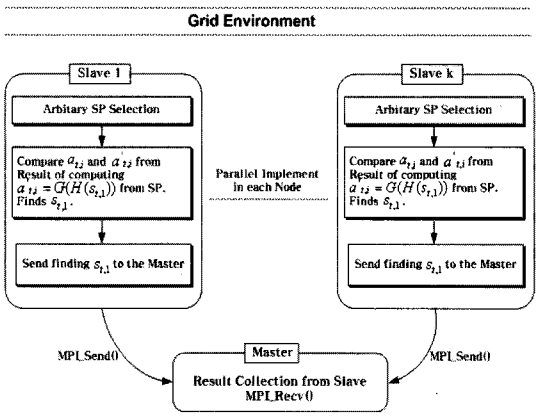

그림 7. 마스터와 슬레이브의 역할 및 통신 방법
Fig. 7. Communication Method and Role of Master and Slave

RFID 태그의 샘플 데이터는 16byte의 시드 값을 갖는 임의의 데이터를 생성하여 사용한다. 전체 태그의 개수 m 즉 SP(Startpoints)는 1000, 2000, 3000, 4000개로 증가시키면서 실험하였고 최대 Hash-Chain의 길이 n은 1000으로 실험하였다. 성능평가 방법은 Hash-Chain 길이(n)와 태그판별개수는 고정된 상태에서 SP 개수와 노드 수를 증가하면서 태그판별시간을 평가하였다. 단 테스트는 각각 100번씩 수행하여 평균으로 산출하였다.

4.2 보안 분석

기존의 기법들과 제안한 방법에 대해 필수 보안 요건인 기밀성, 불구분성, 전방보안성 등의 보장 여부는 아래와 같이 보장한다.

기밀성은 태그가 송신하는 값은 일방향 해시 함수 G의 계산 결과 값 $a_{t,i}$는 물품에 대한 어떠한 정보도 가지고 있지 않기 때문에 문제가 되지 않는다. 또한 태그의 내부에도 물품에 대한 직접적인 정보가 들어있지 않기 때문에 공격자가 태그의 송신정보를 통해서 물품에 대한 유용한 정보를 획득할 가능성은 없다. 따라서 본 제안 기법에서는 RFID 시스템에서 프라이버스 보호를 위한 필수 요건 중 하나인 기밀성을 만족한다.

불구분성은 공격자가 태그의 다음 단계의 송신 값 $a_{t,i+1}$을 현재까지의 모든 송신 값 $a_{t,i}$를 가지고 예측하기 불가능하다. 이는 일방향 해시 함수 G의 입력 값 $s_{t,i+1}$을 공격자가 알아내기는 어렵기 때문이다. $s_{t,i+1}$을 예측할 수 있으려면 $s_{t,i}$ 값을 $a_{t,i}$알아내야 하는데 $s_{t,i}$ 값은 외부로 전송하지 않고, G 해시 함수를 거쳐 계산된 결과를 전송하기 때문에 일방향 해시 함수의 특성상 $s_{t,i}$를 예측하는 것은 계산적으로 불가능하다. 따라서 제안한 기법에서는 RFID 시스템에서 프라이버시 보호를 위한 필수 요건 중 하나인 불구분성을 만족한다.

전방보안성은 태그가 물리적인 공격을 당하는 경우에는 현재의 시드 값 $s_{t,i}$가 노출이 되지만, 공격자는 $s_{t,j}$ 값을 알아낼 수 없다. 이는 일방향 해시 함수 H의 일방향성을 근거로 한다. 또한, 공격자는 태그가 전송하는 값 $a_{t,i}$를 알아내었다하더라도 G 해시 함수의 일방향성을 근거로 $s_{t,i}$를 알아내기는 어렵다. 따라서 본 제안 기법에서는 RFID 시스템에서 프라이버시 보호를 위한 필수 요건 중 하나인 전방보안성을 만족한다.

4.2 실험 결과

SP 개수는 4000개로 고정하여 각 노드에 균등분할과 SP분할 알고리즘을 적용하였을 경우 각 노드의 태그판별시간을 성능을 측정하였다. 그 결과를 단일 노드와 비교하면 균등분할과 SP분할 알고리즘은 노드 수가 2개일 때 7%, 7%, 노드 수가 3개일 때 30%, 34%, 노드 수가 4개일 때 46%, 55%로 성능이 향상 되었다. 이는 단일 노드에 비해서 노드 수가 증가할수록 큰 폭으로 태그 판별 시간이 단축됨을 확인하였다. 또한 Hash-Chain 길이(n)와 태그판별개수는 고정된 상태에서 SP 개수를 증가하면서 노드 수 별로 태그판별시간을 평가한다. SP 개수에 따른 수행결과는 그림 8과 같다. 이 때 SP 개수와 노드 수가 증가할수록 큰 폭으로 태그처리시간이 감소되는 것을 구현 결과를 통해 비교하였다.

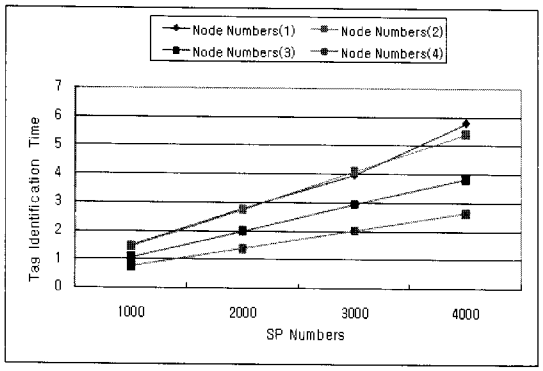

그림 8. SP들의 총수에 따른 성능 비교
Fig. 8. Performance Comparison by the Total Number of SPs

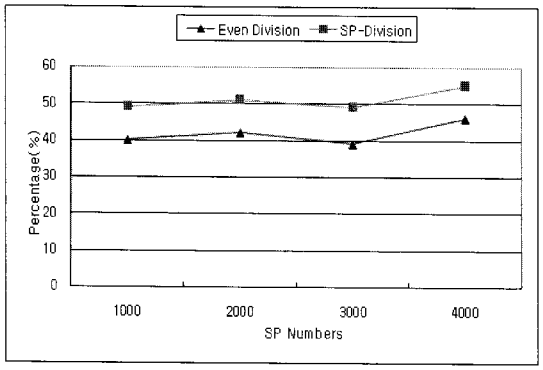

그림 9. SP들의 총수에 따른 성능 향상률
Fig. 9. Performance Elevation Percentage by the Total Number of SPs

그림 9도 같은 조건하에서 노드 수를 4로 고정하여 측정한 결과를 단일 노드와 균등분할, SP분할을 비교하면 SP들의 총수가 1000개 일 때 40%, 49%, 2000개 일 때 42%, 51%, 3000개 일 때 39%, 49%, 4000개 일 때 46%, 55%가 향상되었다. SP들의 총수가 계속 증가하고 노드 수가 더욱 증가한다면 SP 분할 알고리즘을 사용하면 단일 노드나 균등분할을 사용할 때보다 태그를 판별하는데 걸리는 시간은 더욱 감소될 것이라 사료된다. 즉 그리드 환경에서 대용량 처리의 최대 효과를 볼 수 있음을 알 수 있었다.

V. 결론

본 논문에서는 프라이버시 보호를 위해 RFID 태그에 적용한 Hash-Chain 기법의 병행성을 분석하여 그리드 환경으로 이식하였다. 또한 각 노드의 성능 측정 결과를 이용하여 SP들을 분할하는 SP분할 알고리즘을 제안하여 구현하였다.

RFID 시스템에서 프라이버시 침해 문제를 해결하기 위해 사용한 Hash-Chain 기법은 백엔드 서버에서의 많은 계산량으로 인해 태그 판별 시간에 많이 걸리는 문제점 때문에 실제로 활용되기에는 어려움이 있다. 본 논문에서는 이러한 문제의 해결을 위해 먼저 Hash-Chains이 서로 독립적으로 계산이 가능하다는 점을 이용하였다. 또한 태그를 판별하는 과정도 서로 다른 SP에 대해서 전혀 종속성이 존재하지 않는 점을 이용하여 그리드 환경으로 이식하였다. 기본적으로 k 개의 노드가 있을 경우, 각 노드에 SP들을 균등하게 분할하여 적용한 방법은 이질적인 시스템으로 구성되는 그리드 환경에서는 최적화된 성능을 얻을 수 없었다. 이를 해결하기 위한 방법으로 노드의 성능에 따라 SP들을 다양하게 분할하는 SP분할 알고리즘을 제안하였다.

제안한 방법에서는 기존의 단일 백엔드 서버에서의 작업이 그대로 그리드 환경으로 이관되어 처리되는데, 먼저 태그에 적용한 Hash-Chain 기법의 병행성 분석을 통해 분할된 SP들을 병행처리가 가능하도록 그리드에 이식하였다. 기본적으로 k 개의 노드가 있을 경우, 각 노드에 SP들을 균등하게 분할하는 방법은 이질적인 시스템으로 구성되는 그리드 환경의 특성으로 인한 효율성 저하 문제가 발생한다. 이를 해결하기 위해 Hash-Chain 계산 테이블을 생성하는 프로그램을 작성하여 각 노드의 성능을 측정한 후 그 결과를 이용하여 SP들을 분할하는 SP 분할 알고리즘을 제안하였다. 또한 그리드 환경의 태그 판별처리 모델을 구축하고 SP 분할 알고리즘을 적용하여 구현하였다.

구현 결과 Hash-Chain 길이 1000, 노드 수 4로 고정된 상태에서 측정한 결과를 단일 노드와 균등분할, SP분할을 비교하면 SP들의 총수가 1000개 일 때 40%, 49%, 2000개 일 때 42%, 51%, 3000개 일 때 39%, 49%, 4000개 일 때 46%, 55%가 향상되었다. SP들의 총수가 계속 증가하고 노드 수가 더욱 증가한다면 SP 분할 알고리즘을 사용하면 단일 노드나 균등분할을 사용할 때보다 태그를

판별하는데 걸리는 시간은 더욱 감소될 것으로 사료된다. 즉 그리드 환경에서 대용량 처리의 최대 효과를 볼 수 있음을 알 수 있었다.

본 논문에서는 안전한 프라이버시 보호를 위해서 RFID 태그에 Hash-Chain 기법을 이용하였으며, 태그를 판별하기 위해서 그리드 환경에서 최적화된 성능을 얻기 위하여 Hash-Chain 계산 테이블을 생성하는 프로그램을 작성하여 각 노드의 성능을 측정한 후 그 결과를 이용하여 SP를 분할하는 SP분할 알고리즘을 제안하였다. 이 알고리즘을 구현함으로써 균등알고리즘보다 태그 처리 시간이 감소함을 보였다. 향후 연구 방향으로는 많은 실험을 통해서 시스템을 최적화시키는 연구도 필요하다.

참고문헌

[1] Willam P. Walsh, Research and application of RFID Technology to enhance aviation security, IEEE, 2000.

[2] S. Sarma, S. Weis, D. Engles, "RFID Systems and Security and Privacy Implication," In CHES, vol. 2523 of LNCS, pp. 454-469, August 2002.

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

[4] Hyung-Jun Kim, Sung-up Jo, Yong-won Kwon, So-Hyun Ryu, Yong-je Woo, Chang-Sung Jeong, and hyoungwoo Park, "Fast Parallel Algorithm for Volume Rendering and Its Experiment on Computational Grid", ICCS 2003, lNCS 2657, pp. 610-618, 2003.

[5] I. Foster and C. kesselman (eds.) "The Grid: Blueprint for a new Computing Infrastructure," Morgan Kaufman Publishers, 1998.

[6] MPICH-G2, http://www3.niu.edu/mpi

저자소개

신명숙(Myeong-Sook Shin)

2005년 조선대학교 전자정보공과대학 컴퓨터공학과 (공학박사)
2007년 조선대학교 전자정보공과대학 컴퓨터공학과 겸임교수(현)

※ 관심분야 : 운영체제, 유비쿼터스컴퓨팅, 정보보호

안성수(Seong-Soo Ahn)

1986년 조선대학교 수학과(이학사)
1988년 조선대학교 대학원 수학과 (이학석사)
1993년 조선대학교 대학원 수학과 (이학박사)
1992년 동신대학교 컴퓨터공학과 교수(현)

※ 관심분야 : 운영체제, 정보보호, 유비쿼터스컴퓨팅

이 준(Joon Lee)

1979년 조선대학교 전자공학과 (공학사)
1981년 조선대학교 대학원 전자공학과 (공학석사)
1997년 숭실대학교 대학원 전자계산학과 (공학박사)
1982년 조선대학교 전자정보공과대학 컴퓨터공학부 교수(현)

※ 관심분야 : 운영체제, 정보보호, 유비쿼터스컴퓨팅