그룹상호배제 기반의 안전한 분산프로토콜

박 재 혁, 김 광 조

국제정보보호기술연구소, 한국정보통신대학원대학교

A Secure Distributed Protocol based on Group Mutual Exclusion

JaeHyrk Park, Kwangjo Kim

International Research center for Information Security(IRIS)

Information and Communication Univ.(ICU), Korea

요 약

이 논문에서는 Manabe[2]에 의해 제안된 그룹상호배제를 위한 쿼럼(Quorum)기반의 알고리즘을 바탕으로 암호 기법을 이용한 보다 안전한 분산 알고리즘에 대하여 논한다. 그룹 상호배제는 하나의 리소스를 같은 그룹 내의 모든 프로세스에 의해서 공유되도록 할 수 있는 상호배제의 일반화이다[1][4]. 하지만, 다른 그룹의 프로세스들은 상호 배타적인 방법으로 하나의 리소스를 사용하도록 요청된다. 즉, 다른 그룹의 프로세스들은 이미 임계영역에 있는 프로세스가 그 리소스에 대한 사용이 끝난 후 임계영역에 들어갈 수 있다.

분산 컴퓨팅 분야에서 계속적으로 제안된 알고리즘은 실제 개방된 인터넷상에서 각 프로세스들 간의 안전한 통신이 이루어져야 함에도 불구하고 프로세스들 간의 상호배제에만 초점을 맞춤으로서 안전성을 전혀 고려하지 않고 있다. 이 논문에서는 분산알고리즘에 암호학적 기법을 적용한 안전한 분산 알고리즘을 제시한다.

I. 서론

어떻게 하면 분산된 환경에서 여러 개의 프로세스들이 충돌 없이 효율적으로 리소스에 접근 하는지와 관련된 분산 알고리즘에 대한 연구는 분산 컴퓨팅 분야에서 오래 전부터 이루어져왔고, 쿼럼(Quorum)기반의 그룹단위의 분산알고리즘은 2001년 Joung에 의해 처음으로 제시되었다[1]. 이후, Joung의 알고리즘에 불필요한 절차를 제거한 그룹분산알고리즘이 제안되었다[2][6].

정보보호분야에서는 분산 환경에서 프로세스들의 리소스에 안전한 접근을 위해 비밀분산방법이 이용되어왔지만 프로세스들의 충돌여부는 고려하지 않았었다. 단지, 1998년 Naor에 의해 분산 컴퓨팅 분야에 쓰이는 쿼럼 시스템을 이용하여 프로세스간의 장애 허용(fault-tolerant)을 보장하는 안전한 분산 프로토콜이 제안되었다[3].

분산된 환경에서 프로세스들이 같은 리소스를 동시에 요청할 수 있기 때문에 충돌여부는 중요하다고 할 수 있다. 또한 공개된 인터넷환경에서 프로세스간의 안전한 통신 역시 중요하다고 할 수 있다. 이 논문에서는 실제 분산된 인터넷 환경에서 위의 두 가지 큰 특성을 만족하는 분산 알고리즘을 제시한다.

본문의 구성은 다음과 같다.

제 2장에서는 분산 컴퓨팅 분야의 그룹 상호배제에 대한 소개와, 쿼럼 시스템과 비밀분산방법을 이용하여 제시된 기존의 분산프로토콜을 설명하고, 제 3장에서는 새로운 분산 프로토콜을 제안하며, 마지막으로 제 4장에서 결론을 맺는다.

II. 관련연구

1. 그룹 상호 배제

1) 기본 모델

그룹 상호배제는 2001년 Joung에 의해 모델화

되었으며, 이것은 그 동안 연구되어온 상호배제의 일반화이다[1][4]. 그룹 상호배제에서 하나의 프로세스가 임계영역에 들어가기 위해서(리소스를 사용하기 위해서) 다른 프로세스들에게 세션(원하는 데이터 부분)을 요청한다. 만약 다른 프로세스가 이미 사용되어지고 있는 같은 세션을 요청한다면, 동시에 임계영역에 들어갈 수 있다. 실제 예로, 여러개의 프로세스들에 의해 공유된 CD주크박스가 있다고 해보자. 수많은 프로세스들이 동시에 로드된 CD에 접근할 수 있다. 현재 로드된 CD가 아닌 다른 CD에 접근을 요청한 프로세스들은 기다려야 한다. 즉, 여기서 로드된 CD를 요청한 프로세스들은 같은 그룹에 속한 것이고, 그러므로, 동시에 로드된 CD에 접근할 수 있어 효율성을 높일 수 있다[4].

2) 기본 요구사항

그룹의 상호배제를 위해서는 다음과 같은 조건을 만족해야 한다[1][2][6].

상호배제(Mutual exclusion)

어떠한 경우에도, 다른 그룹의 프로세스들이 동시에 임계영역에 들어갈 수 없다.

굶주림 제거(Starvation freeness)

임계영역을 들어가기를 원하는 프로세스들은 결국에는 들어갈 수 있다.

같은 그룹 프로세스들이 계속해서 임계영역에 들어간다면, 다른 그룹의 프로세스들은 계속 기다려야 하는 굶주림상태에 빠질 수 있다. 그러므로 그런 굶주림상태를 피할 수 있는 메커니즘이 이루어져야한다

교착상태 제거(deadlock freeness)

다른 데이터에 접근하기를 원하는 각각의 사용자가 영원히 서로의 데이터의 접근을 기다려서는 안 된다.

3) 쿼럼 기반의 알고리즘

Joung[1]에 의해 정의된 m그룹 쿼럼 시스템을 보자. P={1,...N} 은 m개의 그룹에 소속된 노드들의 수이다.

m-그룹 쿼럼 시스템

프로세스P의 집합들로 구성된 $A=\{C_1,...,C_m\}$ 은 m개의 Coterie(하나의 Coteriie는 여러개의 쿼럼들의 집합)로 구성되어 있다. 아래와 같은 특성이 지켜져야 한다.

Non-emptiness

$$\forall 1 \le i \le m,\ Q_i \neq \emptyset$$

교차 특성(Intersection Property)

$$\forall 1 \le i, j \le m, i \neq j,\\ \forall\ Q_1 \in C_i, \forall\ Q_2 \in C_j\\ \Rightarrow Q_1 \bigcap Q_2 \neq \emptyset$$

최소성(Minimality)

$$\forall 1 \le i \le m, \forall\ Q_1,\ Q_2 \in C_i\\ Q_1 \neq Q_2 \Rightarrow Q_1 \not\subseteq Q_2$$

프로세스들로 구성된 하나의 쿼럼 멤버는 한번에 하나의 프로세스에게만 권한을 준다. 교차 특성에 따라, 각각의 다른 그룹(같은 리소스를 원하면 같은 그룹으로 구성된다)의 두 개의 프로세스가 동시에 임계영역에 들어갈 수 없다.

2. 비밀분산방법을 이용한 분산프로토콜

1) 기본모델

정보보호분야에서 쿼럼 기반의 다자간 프로토콜에 대한 연구[3][5]는 그동안 많이 이루어지지 않았다. 하지만, 정보보호분야에서 분산 컴퓨팅에 쓰이는 쿼럼 시스템을 이용하여 프로세스간의 장애허용이 보장되도록 하는 프로토콜이 제시되었다[3][5].

[3]에서는 쿼럼 시스템을 이용하여 안전하게 데이터베이스에 접근을 하기 위한 방법이 제안되었다. 엑세스 서버들과 데이터서버는 따로 분리되어 있다. 데이터서버는 저장된 모든 정보가 암호화되기 때문에 물리적으로 보호될 필요가 없다. 데이터서비스를 원하는 클라이언트는 하나의 쿼럼의 모든 프로세스들로부터 권한을 얻어야 비밀분산방법을 이용하여 비밀 값을 복원할 수 있다. 악의적인 공격자가 장애가 발생되는 서버로부터 비밀정보를 얻지 못하도록 할 뿐만 아니라, 프로세스간의 일관성을 유지하기 위하여 분산 컴퓨팅 분야에 쓰이는 쿼럼 시스템을 이용한다. 실제 제안된 프로토콜은 다음과 같다. 클라이언트 측의 사용자가 데이터에 대한 접근을 원할 때 그는 엑세스 서버들로 구성된 하나의 쿼럼 안의 모든 서버에게 권한 요청을 한다. 요청을 받은 각각의 서버는 등록된 사용자의 여부를 확인한 후 부분적 비밀 값을 생성하여 요청한 클라이언트에게 값을 전송한다. 요청한 모든 서버로부터 받은 부분적 비밀 값으로부터 비밀 키를 복원한다. 이때 각각의 서버들 사이에는 어떠한 협력도 없다.

2) 문제점

[1]에서 제시된 프로토콜은 쿼럼 시스템과 비밀분산방법을 이용하여 안전한 프로토콜을 제시하였다. 하지만, 프로세스간의 충돌여부를 고려하는 분산알고리즘을 기반으로 하여 제시된 상호배제 알고리즘[1][2][4][6]이 아니므로, 같은 데이터를 원하는 프로세스 간에 충돌이 일어났을 때 해결할 방법이 없다. 프로세스간의 충돌 없는 데이터 공유와 안전성 두 가지 모두에 초점을 맞추어 상호배제 알고리즘을 기반으로 한 안전한 프로토콜을 제안한다.

III. 제안방법

1. 가정

일반적으로 본 연구를 적용하기 위한 상황으로 다음을 가정한다.

-분산된 인터넷 환경이므로, 각각의 서버들끼리는 어떠한 협력도 이루어지지 않는다.

-클라이언트 측의 각각의 사용자들은 서버들과 안전하고 인증된 채널을 이용하여 통신을 한다.

-서버들은 신뢰할만하지만, 항상 사용자를 위한 최신정보를 갖고 있지 못할 수 있다. 분산 환경에서 모든 서버가 최신의 정보로 업 데이트 하는 것은 어렵다.

2. 요구사항

제안된 프로토콜은 다음과 같은 요구사항을 만족한다.

기밀성 : 하나의 쿼럼 내의 부분적인 서버들로부터 키 값을 복구하기 위한 어떠한 부분적 비밀정보도 얻을 수 없다.

인증성 : 데이터에 접근하기 전에 사용자는 한 쿼럼의 모든 서버로부터 인증여부를 확인 받아야 한다.

공평성 : 데이터에 대한 사용자의 접근허가는 모두에게 적절하게 주어져야한다.

위조불가능성 : k개의 서버보다 더 적은 것으로부터 완벽한 키 정보를 위조할 수 없다.

정확성 :비록 어떤 악의적 일이 발생해도 적법한 사용자는 f()로부터 정확한 값을 얻을 수 있다.

장애 허용성 : 각각 서버의 장애가 발생해도 높은 확률을 갖고 서비스는 계속 이루어진다.

효율성 : 같은 데이터를 원하는 사용자들은 동시에 데이터베이스에 접근 가능하다.

그룹 상호 배제 : 만약 같은 그룹을 선택하지 않았다면 동시에 두 프로세스가 임계영역에 들어갈 수 없다.

3. 제안된 프로토콜

제안된 프로토콜은 Manabe의 알고리즘[2]을 기반으로 하여 수정된 것이다. 안전한 비밀 키 전송을 위해 비밀분산방법이 적용되었고, 알고리즘은 크게 두 부분으로 요청을 하는 클라이언트 측의 요청 프로세스 알고리즘과 권한을 주는 서버 측의 서버 프로세스 알고리즘으로 구성된다. 전체 알고리즘은 별첨을 참고하기 바라며, 여기서는 프로토콜의 처리흐름에 대해 언급한다.

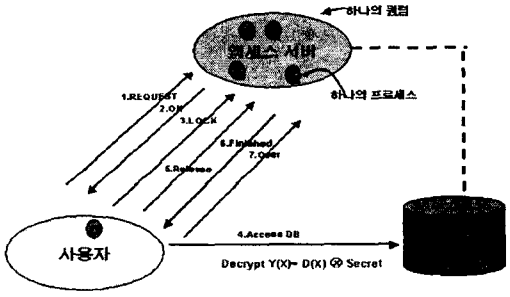

그림 1 : 제안된 프로토콜

위의 그림은 수많은 엑세스 서버들 중 하나의 선택된 쿼럼과 특정데이터를 원하는 사용자 p의 프로토콜의 부분적 그림을 보여준다.

초기화단계

사용자 p는 우선 원하는 데이터X에 대해 돈을 지불하고 등록을 한다. 사용자의 ID정보와 데이터 X의 리스트가 각각의 엑세스 서버 프로세스 q에 저장된다.

요청 프로세스(RequestProcess)

1)사용자 p가 데이터를 원할 때 "Request (G,ID,X)"를 하나의 쿼럼의 모든 서버프로세스에게 보낸다. 여기서 G는 프로세스P가 어느 그룹에 속하는지에 대한 집합이며, 쿼럼은 몇 개의 엑세스 서버 프로세스집합이다.

2)임계영역에 들어가기 위해서는(원하는 데이터를 얻기 위해서는) 세 가지 단계가 있다.

-쿼럼의 모든 서버 프로세스 q들로부터 "OK"메시지(permission)를 받았을 때 p는 자기가 속한

그룹들 중 원하는 그룹을 랜덤하게 선택하고 키복원함수 h를 이용하여 비밀 키 값을 얻는다.

-원하는 데이터에 접근하기 전에 "Lock(g)"메시지를 선택된 쿼럼의 모든 프로세스들에게 보낸 후 임계영역에 들어가 데이터를 복호화 해준다.

-p가 "Enter(g)"를 쿼럼의 어떤 프로세스로부터 받았을 때, p는 자신이 속하는 그룹으로 g를 선택한 후 임계영역에 들어간다.

3)임계영역으로부터 나갈 때

-"OK"에 의해서 임계영역에 들어갔었을 때는, 사용자p는 쿼럼의 모든 엑세스 서버에게 "Release"메시지를 보낸다. 사용자p가 "Finished"를 받으면 그것은 모든 엑세스 서버에게 "Over"를 보낸다.

-"Enter"에 의해 임계영역에 들어갔었을 때는, p는 "NoNeed"를 "Enter"가 도착한 프로세스에게 보낸다.

서버 프로세스(ManagerProcess)

1)사용자로부터 "Request(G,ID,X)"를 받았을 때 받은 ID와 데이터X로부터 적법한 사용자인지 체크한다.

2)적법한 사용자가 맞고 엑세스 서버가 다른 사용자에게 권한을 안 주었거나 혹은 다른 사용자가 같은 그룹이면 비밀분산방법을 통해 비밀 키의 한 부분 정보를 계산한다.

3)사용자로부터 "Lock(g)"메시지를 받았을 때, 서버프로세스는 같은 그룹에 포함되는 프로세스요청에 대해 "Enter(g)"보내어 임계영역에 들어오도록 해준다.

4)"Release"메시지를 받았을 때, "NoNeed"메시지를 받았을 때와 "Over"를 받았을 때는 Manabe가 제안한 그룹상호배제 알고리즘의 절차[2]와 같다.

4. 알고리즘 분석

위와 같이 암호학적 기법을 적용한 안전한 분산 알고리즘을 제시하였다. 분산 컴퓨팅 분야에서 쓰이는 그룹상호배제 알고리즘의 특성은 그대로 유지하며, 데이터에 대한 기밀성, 인증성과 위조불가능성을 만족하는 안전한 알고리즘을 제안하였다. 아래의 표에서 보듯이 [1][2]은 순수한 그룹상호배제 알고리즘이 만족하는 요구사항이며 [3]은 비밀분산방법과 쿼럼 시스템의 개념을 도입하여 제시된 알고리즘이다. 우리가 제안한 새로운 알고리즘은 그룹상호배제 알고리즘을 기반으로 암호적 요소를 적용하여 프로세스간의 안전하고 충돌이 없는 알고리즘을 제안하였다.

(O : Yes, X :No)

분야	요구사항	[1]YJ01	[2]MP03	[3]NW98	제안된 방법
정보보호	기밀성	X	X	O	O
	인증성	X	X	O	O
	공평성	O	O	X	O
	위조 불가능성	X	X	O	O
	효율성	O	O	X	O
	정에 허용성	O	O	O	O
	일관성	O	O	O	O
분산 알고리즘	상호 배제	O	O	X	O
	공주병 제거	O	O	X	O
	교착상태 제거	O	O	X	O

표1 : 알고리즘 간 비교표

IV. 결론 및 향후 과제

본 논문에서 현재의 개방화되고 분산화 된 인터넷 환경의 특성을 고려하여, 안전하고 효율성 있는 분산알고리즘을 제안하였다.

향후, 분산 컴퓨팅 분야에서도 위에 제시된 알고리즘과 같이 안전성도 고려한 상호배제 알고리즘이 활발히 제시되어야 할 것이다.

참고문헌

[1] Y.-J.Joung, "Quorum-Based Algorithms for Group Mutual Exclusion", DISC'2001, LNCS 2180, pp16-32, 2001

[2] Y.Manabe and JaeHyrk Park ,"A Quorum-based group mutual exclusion algorithm without unnecessary blocking", submitted to SRDS'2003

[3] M.Naor and A.Wool,"Access Control and Signatures via Quorum Secret Sharing", IEEE Transactions on Parallel and Distributed Systems, pp909-922, 1998

[4] V.Hadzilacos, "A note on group mutual exclusion", In Proc. 20th PODC, 2001

[5] D.Beaver and A.Wool,"Quorum-based secure multi-party computation",Advances in Cryptology-EUROCRYPT'98, LNCS1403, pp375-390,1998

[6] JaeHyrk Park, Y.Manabe and Kwangjo Kim,"Quorum-based Algorithm using Group Choice", CISC'2002

별첨

```
Program RequestingProcess(p:process);
var Rstatus = wait : status of request;
Q = Ψ : set of process; /* quorum */
K : set of process;     /* reply is received */
G : set of group; /* current group set */
ID : process D
X : data part that user want
T: set of partial secret value

When p(group set is G) wants to enter CS;
begin
  Rstatus := wait;
  Select arbitrary Q from coterie;
  K :=Ψ;
  T :=Ψ;
  send "Request(G,ID,X)" to all q∈Q;
end; /*end of request initiation.  */

At arrival of "OK" from q;
begin
 if Rstatus = wait then begin
  K:=K∪{q} and T:=T∪{s} ; /*collect s from each q */
  if K=Q then /*tnat means collecting secret value finish*/
   begin        /*can enter CS */
   secret = h({si from T}) /*h : reconstruction function */
   select arbitrary g∈G;
   send "Lock(g)" to all q∈Q;
   Rstatus := in;
/*in the CS, access data that user want */
   y(x) = D(x) XOR secret
   Rstatus := out;
   send "Release" to all q∈Q;
   K :=Ψ;  /* waits for "Finished */
  end           /* end of Q=K */
 end            /* end of Rstatus= wait*/
end;            /* end of arrival "OK" */

At arrival of "Enter(g)" from q;
begin
 if Rstatus= wait then begin
   send "NoNeed" to all r∈Q-{q};
   Rstatus := in;
   y(x) =D(x) XOR secret
   Rstatus := out;
   send "Noneed" to q;
end; /*end of Rstatus= wait */
end; /* end of arrival "Enter" */

At arrival of "Cancel" from q;
begin
 if Rstatus= wait then begin
  K:= K-{q};
send "Cancelled" to q;
end; /* end of Rstatus=wait */
end; /* end of arrival "Cancel" */

At arrival of "Finished" from q;
begin
  K:= K∪{q};
  if K=Q then send "Over" to all q∈Q;
end;  /*end of arrival "Finished" */

Program ManagerProcess(q:process);
var status = vacant: status;
  group: group;  /*current group */
  Que = null: priority queue of requests;
  waiting= null: process; /* waits "Lock" */
  sentok = null: process  /* 'OK' is sent */
  Que = null : priority queue of requests;
  using = null : set of processes;

At arrival of "Request(G,ID,X)" from p:
If ID∈ registration_list  /*check authorization */
begin
  Insert the request to Que;
  /* assume that Que[i] be the position */
  Que[i].status := wait;
  Que[i].pr := p;
  Que[i].G := G;
  if status = vacant then begin
/* generate o with encryption random function Keyo(x)*/
o =Keyo(x);
/* generate pseudo-random string with a private seed rr*/
r=Rrr(x XOR ID);
/* computes its share of the key, si using the SSS */
si= Πi(o,r)
send"OK" to p;
sentok := p;
Que[i].status := waitlock;
status := waitlock;
end /* end of vacant */

else if status = locked then begin
 if group∈G and Que[1].status = enter
/*Que[1] : highest priority request */
then begin
o =Keyo(x);
r=Rrr(x XOR ID);
si= Πi(o,r)
send "Enter(group)" to p;
using := using +{p}
Que[i].status := enter;
end
end /* end of locked */

else if status=waitlock then begin
 if Que[i] is highest priority then begin
   send "Cancel" to process sentok;
   /* assume Que[k].pr= sentok */
   Que[k].status:=waitcancel;
   status:= waitcancel;
 end
end /* end of waitlock */
end; /* end of "Request" arrival */
```

```
At arrival of"Lock(g)" from p(p=Que[i].pr);
begin
  using={p}
  Que[i].status := enter;
  status := locked;
  group:= g;
  if Que[1].status= enter or g∈Que[1].G
   then begin /*Que[1] can enter CS */
    for every request Que[k](k≠i) such that
      g∈Que[k].G do begin
       send "Enter(g)" to Que[k].pr;
       Que[k].status := enter;
       using := using+{Que[k].pr};
     end; /* end of do */
  end;
end;  /* end of arrival "Lock"

At arrival of "Release" from p(p=Que[i].pr);
begin
  status := releasing;
  remove entry Que[i];
  using := using -{p};
  if using=Ψ then send "Finished" to p;
end;

At arrival of "NoNeed" from p(p=Que[i].pr):
begin
  remove entry Que[i];
  if p= sentok then NewChance
   else if p∈using theen begin
    using:=using-{p};
    if using = Ψ then send "Finished" to sentok;
  end
end;

At arrival of "Cancelld" from p(p=Que[i].pr);
begin
  Que[].status:= wait;
  SendOk;
end;

At arrival of "Over" from p(p=Que[i].pr):
 SendOk;

procedure SendOk; /* permission released. */
begin
  if Que is not empty then begin
/* Que[1]: highest priority request */
Send "Ok" to Que[i].pr;
Que[1].status := waitlock;
status:= waitlock;
end /*end of Que is not empty */
else status := vacant;
end;
```