Journal of Korea Multimedia Society Vol. 15, No. 8, August 2012(pp. 1048-1058)

클라우딩 기반에서 클라이언트와 서버간 협상을 위한 자가 조직 저장매체의 DDMPF(Distributed Data Management Protocol using FAT) 설계

이병관[†], 정은희[††], 양승해[†††]

요 약

본 논문은 클라우딩 환경에서 클라이언트와 저장 서버, 검증 서버로 구성하여 자가 조직 저장 매체의 데이터 손실을 방지하고, 보안을 유지하기 위한 DDMPF(Distributed Data Management Protocol using FAT)을 제안한다. DDMPF는 클라우드 컴퓨팅 환경에서 자가 조직 저장 서버를 구축하고, 데이터를 분할하여 저장 서버에 분산 저장함으로써 기존의 클라우딩 저장 매체의 중앙 집중화 문제와 저장 서버 문제로 인한 데이터 손실 문제를 해결하였고, 파일할당테이블을 이용해 분산 저장된 데이터 관리의 효율성도 향상시켰다. 그리고 DDMPF는 저장 서버의 데이터 무결성을 검증 서버가 검증함으로써 데이터의 신뢰성을 향상시키고, 클라이언트의 비밀키와 EC-DH 알고리즘을 이용하여 생성된 시스템 마스터 키로 이중 암호화하여 전송함으로써 보안을 강화시켰다. 또한, 자가 조직 저장 매체를 구성할 때, 검증서버의 개수를 제한하고, 검증요청메시지에 대한 TS(Time Stamp)을 설정함으로써 플러딩 공격 탐지하였고, 검증을 요청할 때마다 새롭게 생성된 nonce 값을 이용하여 재전송 공격을 탐지하도록 하였다.

A DDMPF(Distributed Data Management Protocol using FAT) Design of Self-organized Storage for Negotiation among a Client and Servers based on Clouding

Byung Kwan Lee[†], Eun Hee Jeong[††], Seung Hae Yang[†††]

ABSTRACT

This paper proposes the DDMPF(Distributed Data Management Protocol using FAT) which prevents data loss and keeps the security of self-organized storages by comprising a client, a storage server, and a verification server in clouding environment. The DDMPF builds a self-organized storage server, solves data loss by decentralizing the partitioned data in it in contrast to the centralized problem and the data loss caused by the storage server problems of existing clouding storages, and improves the efficiency of distributed data management with FAT(File Allocation Table). And, the DDMPF improves the reliability of data by a verification server's verifying the data integrity of a storage server, and strengthens the security in double encryption with a client's private key and the system's master key using EC-DH algorithm. Additionally, the DDMPF limits the number of verification servers and detects the flooding attack by setting the TS(Time Stamp) for a verification request message and the replay attack by using the nonce value generated newly, whenever the verification is requested.

Key words: DDMPF, FAT(파일할당테이블), self-organized storage(자가 조직 스토리지), Distributed storage(분산저장), Delegation verification(위임검증), Time stamp(타임스탬프)

※ 교신저자(Corresponding Author) : 정은희, 주소 : 강원도 삼척시 중앙로 1 (245-711), 전화 : 033)570-6646, FAX : 033)574-6640, E-mail : jeongeh@kangwon.ac.kr
접수일 : 2012년 4월 3일, 수정일 : 2012년 5월 23일
완료일 : 2012년 6월 20일
[†] 정회원, 관동대학교 컴퓨터학과 교수
(E-mail : bklee@kwandong.ac.kr)
[††] 정회원, 강원대학교 지역경제학과 부교수
[†††] 정회원, (주)씨디에스 연구소장
(E-mail: ysh@cds.co.kr)
※ 본 논문은 중소기업청에서 지원하는 2011년도 산학연공동기술개발사업(No. 00047189)의 연구수행으로 인한 결과물임을 밝힙니다.

1. 서 론

클라우드 컴퓨팅 환경은 주로 사용자가 데이터를 보유하지 않고 데이터 센터 내에 논리적으로 분리된 저장 공간에서 데이터가 공유되는 환경이기 때문에 데이터 보안의 중요성이 더욱 부각된다. 즉, 클라우드 컴퓨팅 환경의 저장 서비스는 웹 하드처럼 클라이언트가 클라우드 컴퓨팅 서버의 저장장소를 임대해 사용하는 방식으로 데이터와 서비스의 중앙 집중화는 악의적인 공격에 치명적인 결함을 초래할 수 있으며, 서버에 문제가 발생하면 클라이언트는 데이터의 접속이 불가능하거나 데이터의 손실을 초래 할 수 있다.

본 논문에서는 이러한 클라우드 컴퓨팅 환경의 보안 문제점들을 해결하고 데이터의 안전성과 신뢰성을 향상시킬 수 있는 분할, 분산 저장, 위임 검증 기법을 적용하는 DDMPF(Distributed Data Management Protocol using FAT)을 설계한다. DDMPF는 클라우드 컴퓨팅 환경에서 자가 조직 저장 서버를 구축하고, 데이터를 분할하여 분산 저장함으로써 기존 클라우딩 저장 매체의 중앙 집중화 문제와 저장 서버 문제로 인한 데이터 손실 문제를 해결하고자 한다. 또한 클라이언트의 데이터가 분산 저장되어 있는 저장 서버의 데이터 무결성 검증은 검증 서버에 위임하여 데이터의 무결성을 검증함으로써 DDMPF의 데이터 보안 및 데이터 신뢰성을 향상시키고, 클라이언트, 저장서버, 검증서버에 각각의 파일 할당 테이블을 설치하여 데이터의 검색 및 검증 절차 시에 유효성을 증가시키고자 한다..

본 논문의 구성을 살펴보면 2장은 관련 연구, 3장은 DDMPF 보안 설계와 DDMPF의 데이터 관리 설계 그리고 4장은 성능분석, 5장은 결론으로 되어 있다.

2. 관련연구

2.1 자가 조직 보안

P2P와 같이 자가 조직 매체는 분산컴퓨팅 기술을 기반으로 하고 있어 IP Spoofing, 노드 간에 데이터 스트림생성을 통한 불법 수정이나 신분위장(masquerade), 재전송(replay), 그리고 DoS 공격과 같은 보안 취약성을 가진다[1,2]. 이러한 보안 취약성을 해결하기 위한 클라우드 컴퓨팅 환경에서의 보안 제어는 일반적인 IT 환경과 유사하지만, 이것만으로는 클라우드 컴퓨팅의 보안을 해결하는 데는 어려운 문제점이 있다[3]. 여기서는 클라우드 컴퓨팅 환경의 자가 조직이나 인프라 클라우딩 스토리지 서비스에서 사용하는 스토리지 보안 기법들을 관련 연구로 살펴보자.

2.1.1 검색 가능 암호시스템

검색 가능 암호시스템이란 스토리지에 대한 대표적인 보안 기술로서 기존의 암호 기술과 같이 암호화된 정보에 대한 기밀성을 보장하면서 동시에 특정 키워드를 포함하는 정보를 검색할 수 있도록 고안된 암호 기술이다. 이 보안 기술은 암호화된 데이터 외에 검색에 사용할 인덱스(index)를 추가로 생성하여 사용자가 자료를 검색할 때 이 인덱스를 사용한다. 즉, 사용자가 키워드와 비밀키를 사용하여 키워드의 정보를 포함한 트랩도어(trapdoor)를 생성해 서버에 전달하면, 서버는 사용자가 전해준 트랩도어와 저장된 인덱스를 이용하여 검색을 수행하고 결과를 사용자에게 전달한다. 이 과정에서 인덱스와 트랩도어로부터 저장된 자료 또는 사용자가 검색한 키워드에 대한 정보의 유출을 최소화하여 데이터를 보호한다[4,5].

2.1.2 PPDM 기술

데이터 마이닝이 전자상거래나 마케팅과 같은 분야에 활용되면서, 프라이버시 침해 이외에도 개별 회사들 사이에 정보의 노출이 문제가 되었다. 따라서 데이터 소유자의 프라이버시를 침해하지 않으면서 정보를 추출할 때, 정보를 공유도 하고 프라이버시 유지를 위해 개발된 PPDM(Privacy Preserving Data Mining) 기술이 스토리지에 대한 대표적인 보안 기술이다. 하지만, 이 기술 외에 현재 상용되고 있는 데이터 마이닝 소프트웨어에서 제공되는 알고리즘으로는 연관 규칙(association rules), 분류(classification), 순차 패턴(sequential patterns), 군집화(clustering) 등이 있다[5,6].

본 논문에서는 파일 할당 테이블에 저장되어 있는 데이터를 검색하지 않고, 분할파일 ID, 저장서버 ID, 검증서버 ID를 검색어로 사용해서 데이터의 정보를 노출을 방지한다.

2.2 키 교환 프로토콜

키 교환 프로토콜은 인증된 두 사용자가 안전하게 비밀키를 전송하는 방법으로 Diffie-Hellman 키 교환 방식인 Oakley Key Determination Protocol와 IPSec에서 키 교환을 정의하는 ISAKMP(Internet Security Association and Key Management Protocol)이 있고, 이 두 가지 방식의 장점을 통합한 키 교환 방식으로 IKE(Internet Key Exchange)이 있다. EC-DH는 Diffie-Hellman 알고리즘을 타원곡선 위로 옮긴 것으로 X9.63으로 표준화 되어 있다[7-9].

본 논문에서는 Diffie-Hellman 알고리즘을 타원곡선위로 옮긴 것으로 X9.63으로 표준화 되어 있는 EC-DH를 키 교환 프로토콜로 사용하였다[10].

2.3 암호화 알고리즘

데이터를 안전하게 저장하고 전달하는 대표적인 알고리즘은 비밀 키 알고리즘으로 DES(Data Encryption Standard)와 AES(Advanced Encryp-tion Standard), 공개키 암호 알고리즘으로 RSA(Rivest Shamir Adleman)와 ECC(Elliptic Curve Cryptography)가 있다[11,12]. 본 논문에서는 대칭키 알고리즘인 AES을 이용하였으며, 암·복호화 키는 클라이언트 데이터의 경우에는 클라이언트의 비밀키를 사용하였고, DDMPF의 구성요소 간의 데이터 전송할 경우에는 시스템 마스터키를 사용하였다.

2.4 비밀분산기법

비밀분산(Polynomial Secret Sharing)이란 특정 그룹의 참가자들에게 비밀을 분산시키는 것으로 각 참가자에게는 분산정보가 할당된다. 이 비밀은 분산정보가 모아져야만 복구될 수 있으며 각각의 분산정보만으로는 비밀에 대해서 아무것도 알아낼 수가 없다. 이러한 비밀분산은 비밀키의 안전한 저장, 공개키를 이용한 암호화된 백업시스템 구축, 여러 통신경로로 비밀 메시지 전송, 권한 분산 등의 용도로 사용되고 있다[13,14].

본 논문에서 제안하는 DDMPF는 클라이언트가 데이터를 분할한 후 암호화시켜 저장서버에 비밀 분산시켜 저장함으로써, 저장서버에 분산 저장된 데이터가 노출되더라도 실제 데이터에 대한 정보는 알아낼 수 없도록 설계하였다.

3. DDMPF 설계

본 논문에서 제안하는 클라우드 컴퓨팅 환경에서의 자가 조직 저장매체에 대한 안전하고 효율적인 관리를 위한 DDMPF(Distributed Data Management Protocol using FAT)은 데이터를 인프라 클라우딩 환경의 저장매체에 저장할 때 기존의 클라우딩 스토리지 서비스처럼 한 곳에 집중하여 저장하는 것이 아니라, 데이터를 분할하여 분산 저장하고, 저장된 데이터를 제 3자가 검증하도록 함으로써 데이터의 안전성, 무결성, 신뢰도를 향상시키고자 한다.

본 논문에서 제안하는 DDMPF는 데이터의 소유자인 클라이언트와 데이터를 분산 저장할 저장서버, 그리고 저장서버에 저장되어 있는 데이터의 무결성 검증을 위임받은 검증서버로 구성된다[10]. 그림 1은 본 논문에서 설계한 DDMPF의 전체적인 구성과 각 구성요소간의 데이터 저장 및 검증에 대한 흐름을 설명한 것이다.

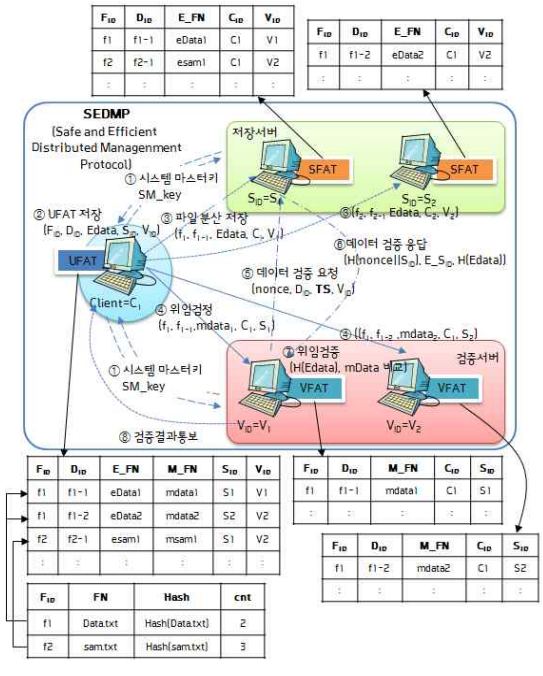


그림 1. DDMPF의 구성요소와 전체적인 흐름도

3.1 DDMPF 보안 설계

3.1.1 시스템 마스터 키 생성

DDMPF는 클라이언트, 저장서버 그리고 검증서

버 간에 사용할 시스템 마스터 키(System Master Key : SM-key)를 생성한다. SM-key는 이 클라이언트, 저장서버, 검증서버 간에 공유하는 키를 말하며, 클라이언트, 저장서버, 검증서버 간의 전송되는 데이터를 암·복호화 할 때 사용하거나, 서로의 신분을 확인할 때 사용된다.

그림 2는 시스템 마스터키인 SM-key를 생성하는 과정을 설명한 것이다. DDMPF에서는 타원곡선 암호 알고리즘을 이용해 암호화에 필요한 파라미터 생성하고, 이 구성요소인 클라이언트, 저장서버, 검증서버에 공개하면, 클라이언트, 저장서버, 그리고 검증서버는 자신의 비밀키와 초기점 P를 곱셈 연산하여 각각의 공개키를 생성한 후 공개한다.

클라이언트는 EC- DH 알고리즘을 이용해 저장서버와 검증서버의 공개키를 자신의 비밀키로 곱셈 연산을 하여 3자간 공유 비밀키가 생성하고, 저장서버는 클라이언트와 검증서버의 공개키에 자신의 비밀키로 곱셈 연산하여 3자간 공유 비밀키가 생성한다. 검증서버 또한 클라이언트와 저장서버의 공개키에 자신의 비밀키로 곱셈 연산하여 3자간 공유 비밀키를 생성한다. DDMPF에서는 이 3자간 공유 비밀키를 시스템 마스터키인 SM-key로 사용한다[10].

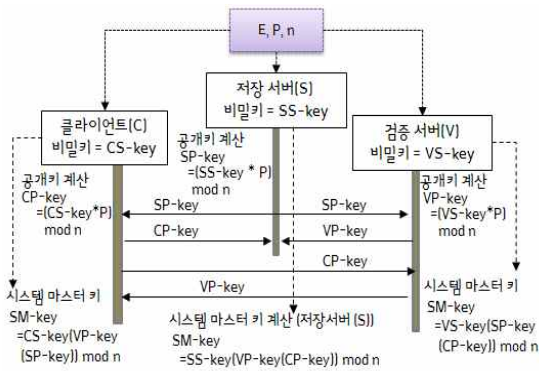


그림 2. 시스템 마스터키 생성 흐름도

3.1.2 데이터 암호화

클라이언트는 데이터를 자신의 비밀키로 암호화하고, 암호화된 데이터를 SM_key로 한번 더 암호화시킨 후 저장 서버에 이중 암호화된 데이터를 전송한다. 이때, 저장 서버는 SM_key로 복호화 시킨 후, 수신된 암호문의 무결성을 검증하는데, 무결성 검증이 유효하다면 데이터 소유자의 비밀키로 암호화된 데이터를 그대로 저장한다. 그리고 SFAT에 데이터 정보와 클라이언트의 ID, 검증 서버 ID를 저장한다. 만약에 무결성 검증이 유효하지 않다면 클라이언트에게 데이터 전송에 에러가 발생하였음을 통지한다.

그림 3은 클라이언트가 데이터를 암호화하여 저장서버에 전송하고, 저장서버는 데이터의 무결성 검증하여 유효한 경우에만 SFAT에 저장하고, 유효하지 않는 경우 에러메시지를 클라이언트에 전송하는 과정을 설명한 것이다.

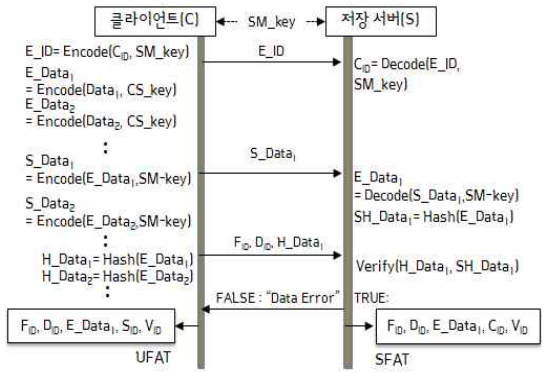


그림 3. 암호화된 데이터 전송 절차

3.1.3 데이터 검증

DDMPF는 시스템 마스터키인 SM-key와 임의의 값인 nonce를 이용해 검증 서버가 저장 서버에 저장되어 있는 데이터의 무결성을 검증하여 그 결과를 클라이언트에게 전송하는 역할을 담당한다. 이때 임의의 값인 nonce는 검증을 요청할 때마다 새로운 값으로 생성되도록 함으로써 재전송 공격을 방지하도록 설계하였고, 요청메시지 생성 시간을 저장서버에 전송함으로써 빈번한 검증 요청 메시지로 인한 플로딩 공격을 방지하도록 설계하였다.

DDMPF의 데이터 검증 절차는 그림 4와 같으며 단계별 처리 과정을 살펴보면 다음과 같다.

[1 단계] 검증 서버는 VFAT에서 검증 서버가 검증을 위임받은 저장 서버를 찾아서 검증 요청 신호를 전송한다. 이때 플로딩 공격을 방지할 목적으로 요청메시지 전송시간을 저장서버에 전송하고, 재응답 공격을 방지할 목적으로 nonce를 랜덤하게 생성하여 전송한다. 그리고 검증 서버의 ID를 SM_key로 암호화하여 저장 서버에 전송한다.

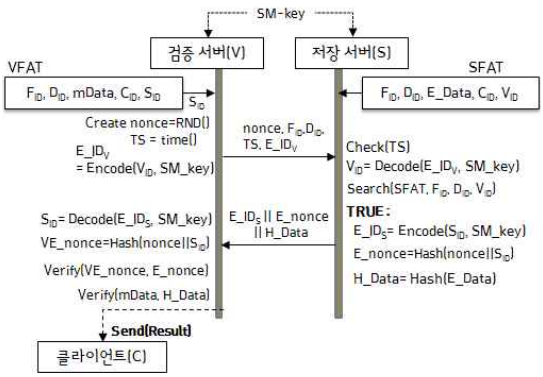


그림 4. 데이터 검증 흐름도

```
RequestMessage(){
    nonce = RND(); // 랜덤한 난수 생성
    TS = time(); // 요청메시지 생성 시간
    SID = FindStorageID(VFAT);
          // VFAT에서 저장서버 ID 찾기
    E_IDV=Encode(VID, SM_Key);
          // 검증서버 ID 암호화
    Send(nonce, FID, DID, TS, E_IDV);
          // 저장서버에 요청메시지 전송
}
```

[2 단계] 저장 서버는 검증 서버의 무결성 요청 메시지의 생성시간을 확인하여 주기적인 검증 요청 메시지인지 혹은 플러딩 공격인지를 확인한다. SM_key로 암호화된 검증 서버의 ID를 복호화 시키고 검증 서버의 ID, 파일 ID, 그리고 분할 파일 ID가 SFAT에 존재하는지를 검사한다. 검증 서버의 ID, 파일 ID, 그리고 분할 파일 ID가 SFAT에 존재한다면, E_Data1을 nonce값을 확인한다. 그리고 SFAT 테이블의 데이터 정보로 저장하고 있던 데이터를 해시한 값과 nonce값을 검증서버에게 전송한다.

```
ResponseMessage(nonce, TS, E_IDv){
    Check(TS); //검증요청메시지 생성시간 확인
    VID = Decode(E_IDV, SM_Key);
          //검증서버 ID 복호화
    IF(FindStorageID(SFAT,FID,DID,VID)){
       //SFAT에서파일ID,분할파일ID,
       //검증서버ID를 찾으면,
       E_IDS=Encode(SID, SM_Key);
       H_nonce = Hash(SID, nonce);
       H_Data = Hash(E_Data);
       Send(E_IDS, H_nonce, H_data);
            //검증서버에 응답메시지 전송
    }else{ //찾지 못하면
       Send("Not Find ID"); //에러 메시지 전송
    }
}
```

[3 단계] 검증 서버는 저장 서버로부터 수신한 암호화된 저장 서버 ID를 SM_key로 복호화 시킨다. 그리고 저장 서버에서 수신한 nonce와 저장 서버 ID를 연접해 해시한 값과 검증 서버가 전송한 nonce값과 복호화된 저장 서버 ID와 연접해 해시한 값과 일치하는지를 먼저 검사한다. 만약에 일치한다면 검증서버가 nonce를 전송한 저장 서버로부터 응답 메시지를 받은 것으로 간주하고, VFAT에서 저장서버 ID를 찾는다. 그리고 VFAT에서 meta 정보를 가져와서 저장 서버가 보낸 meta 정보와 비교한다. 비교한 결과에 따라 검증 서버는 데이터 소유자에게 데이터 무결성 상태를 통보한다.

```
Verification(E_IDs, nonce){
    SID = Decode(E_IDS, SM_Key);
    H_nonceV = Hash(SID || nonce);
    IF(Compare(H_nonceV,H_nonce)){
       IF(FindStorageID(VFAT, SID)){
         //VFAT에서 저장서버 ID 찾으면,
          H_DataV=ReadMeta(VFAT, SID);
             //VFAT에서 meta정보 가져오기
          IF(Compare(H_DataV, H_Data)){
                //데이터 무결성 검증
            Send("Valid Data", UID);
          else
            Send("Invalid Data", UID);
       }else  // VFAT에 저장서버 ID가 없으면,
          Display("Not Find SID");
    }else{     // nonce가 일치하지 않으면
       Display("Not Match nonce");
    }
}
```

[4 단계] 검증 서버는 1, 2, 3단계의 검증 과정을 주기적으로 실시하여 그 결과를 데이터 소유자인 클라이언트에게 통보한다.

3.2 DDMPF 분산 데이터 관리 설계

본 논문에서 제안하는 DDMPF는 클라이언트의 데이터를 분할하여 저장서버에 분산 저장하거나, 데이터를 복제하여 저장서버에 분산 저장함으로써 기존의 중앙집중형 인프라 클라우딩 저장매체 시스템

의 문제점을 해결하고자 한다. 따라서 DDMPF가 데이터를 효율적으로 관리하기 위해서는 분산 저장된 데이터의 위치와 위임 검증할 검증서버의 정보 관리가 필요하다.

그림 5는 DDMPF의 클라이언트 관점의 데이터 관리 절차를 설명한 것이다.

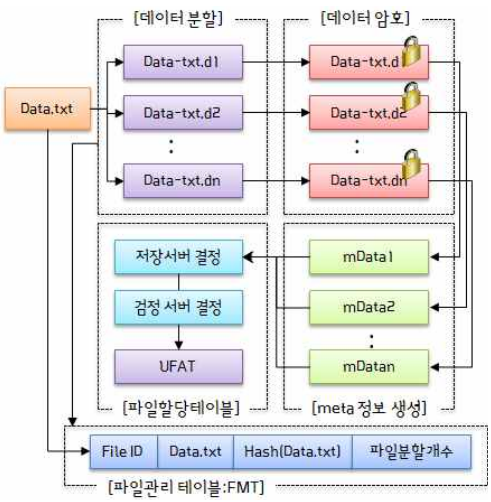


그림 5. 데이터 관리 절차

3.2.1 파일 할당 테이블

DDMPF에서는 데이터 관리의 효율성을 향상시키고자 DDMPF의 구성요소인 클라이언트, 저장서버, 검증서버의 특성에 맞는 파일 할당 테이블을 설계한다. 즉, 파일 할당 테이블은 DDMPF의 각 구성요소에 설치되는 테이블로 각 구성요소의 특성에 따라 데이터 소유자인 클라이언트의 파일 할당 테이블(User File Allocation Table, UFAT), 저장 서버의 파일 할당 테이블(Storage File Allocation Table, SFAT), 검증 서버의 파일 할당 테이블(Verification File Allocation Table, VFAT)로 차별화시켜 설계한다.

(1) 클라이언트의 UFAT

클라이언트에 설치된 UFAT는 클라이언트가 파일을 분할하고 암호화한 후에 저장한 저장서버의 ID와 meta 정보를 전송함으로써 데이터의 검증을 위임한 검증서버의 ID를 저장한다. 또한, 클라이언트는 파일을 분할하여 분산 저장하므로 차후에 분산 저장된 파일을 다시 수집하고, 병합하였을 때, 정확하게 분산 저장된 데이터를 수집하였는지에 대해 확인할 수 있는 분할 파일의 개수와 병합된 파일에 대한 데이터 무결성 검증을 위한 원본 파일에 대한 해시 값이 필요하다.

그림 6은 클라이언트 UFAT의 구조와 클라이언트의 파일관리테이블인 FMT(File Management Table)을 설명한 것이다. 각 파일들은 고유 ID로 일대다 관계로 연결되어 있으며, FMT의 cnt 항목은 클라이언트가 분산 저장된 데이터 파일을 수집하였을 때, 정확하게 수집하였는지 확인할 때 사용하고, 해시값은 병합된 파일의 무결성을 확인하는데 사용한다.

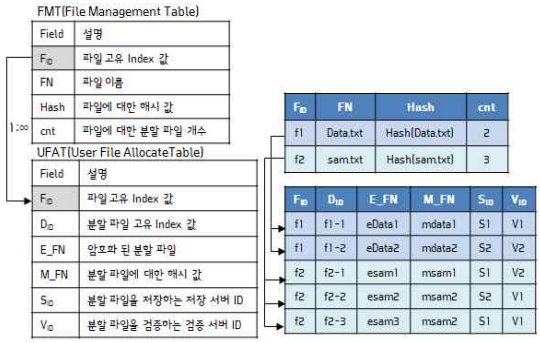


그림 6. 클라이언트의 UFAT와 FMT 구조

(2) 검증서버의 VFAT

검증서버에 설치된 VFAT는 파일의 고유 ID, 분할 파일 고유 ID, 검증서버에 저장된 메타정보의 소유자 ID, 그리고 파일의 무결성을 요청할 저장서버의 ID가 저장한다. 검증서버는 nonce, FID, DID, TS, 그리고 시스템 키로 암호화된 VID로 구성된 검증요청 메시지를 전송한다.

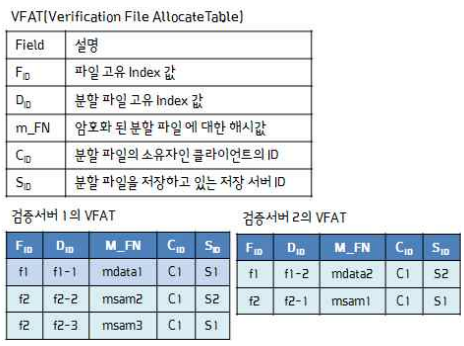


그림 7. 검증서버의 VFAT 구조

(3) 저장서버의 SFAT

저장서버에 설치된 SFAT는 파일의 고유 ID, 분할 파일 고유 ID, 저장서버에 저장된 파일의 소유자 ID, 그리고 저장된 파일의 무결성 검증을 요청하는 검증서버의 ID를 저장한다. 저장서버는 검증서버가 파일의 무결성 검증을 요청할 때, SFAT에서 파일의 고유 ID인 FID, 파일 분할 고유 ID인 DID와 검증서버의 ID인 VID가 일치하는 해당 파일의 해시 값을 검증서버에 전송하게 된다.

SFAT(Storage File AllocateTable)

Field	설명
F_{ID}	파일 고유 Index 값
D_{ID}	분할 파일 고유 Index 값
E_FN	암호화 된 분할 파일
C_{ID}	분할 파일의 소유자인 클라이언트의 ID
V_{ID}	분할 파일을 검증하는 검증 서버 ID

저장서버 1의 SFAT

F_{ID}	D_{ID}	E_FN	C_{ID}	V_{ID}
f1	f1-1	eData1	C1	V1
f2	f2-1	esam1	C1	V2
f2	f2-3	esam3	C1	V1

저장서버 2의 SFAT

F_{ID}	D_{ID}	E_FN	C_{ID}	V_{ID}
f1	f1-2	eData2	C1	V2
f2	f2-2	esam2	C1	V1

그림 8. 저장서버의 SFAT 구조

3.2.2 데이터 분할 모듈

DDMPF에서 클라이언트는 원본 데이터를 일정한 크기로 분할하여 저장서버에 분산시켜 저장하거나, 원본 데이터를 복제하여 저장서버에 분산시켜 저장함으로써 기존의 인프라 클라우딩 저장시스템의 중앙 집중화 문제와 서버 이상에 따른 데이터 손실 문제를 해결하고자 한다.

DDMPF에서 클라이언트는 파일을 분할할 크기를 직접 입력하도록 설계하였으며, 분할된 데이터 이름을 원본 데이터의 이름에 확장자를 덧붙여서 사용하도록 하여 차후에 병합할 때 별도의 데이터 이름을 설정하는 번거로움을 해소하였다. 즉, data.txt라는 원본 데이터가 3개의 파일로 분할된다면, data-txt.d1, data-txt.d2, data- txt.d3라는 파일이름으로 저장하도록 하였으며, data-txt.d1, data-txt.d2, data-txt.d3이 병합될 때에는 자동적으로 파일이름이 data-txt인 것들만 병합하여 data.txt 이름으로 저장하도록 하였다.

그림 9는 데이터 분할의 과정을 설명한 것이다.

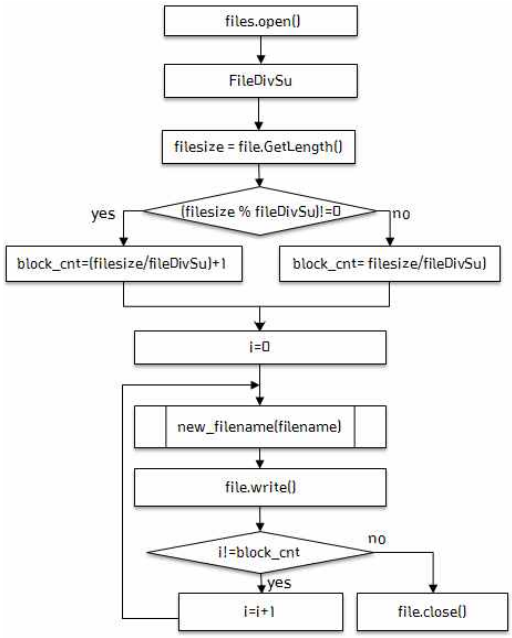


그림 9. 데이터 분할 처리 흐름도

3.2.3 데이터 병합 모듈

클라이언트는 저장 서버에 분산되어 있던 암호화된 데이터를 수집하여 복호화 시킨 후, 완성된 데이터로 병합시킨다. 이때 클라이언트는 분할된 데이터의 무결성을 검증 서버로부터 주기적으로 정보를 수신하도록 설계하였지만, 클라이언트가 병합된 데이터의 무결성을 한번 더 검증하도록 설계함으로써 데이터의 신뢰성을 향상시키고자 하였다.

분할된 데이터들의 이름은 “파일이름_확장자.분할순서”로 저장되어 있으므로, 수집된 데이터들 중에서 병합하고자 하는 파일이름을 선택하면, 폴더 내에 “파일이름_확장자”가 동일한 파일들을 모두 병합하도록 설계하였다.

그림 10은 분할 파일이름이 data-txt인 경우의 데이터 병합과정을 설명한 것이다.

3.2.4 meta 정보 생성

DDMPF에서 클라이언트는 분할된 데이터를 저장서버에 전송하기 전에 자신의 비밀키로 암호화하고, 시스템 마스터키인 SM_key로 한번 더 암호화하여 데이터의 보안을 강화시킨다. 이때, 클라이언트는 암호화된 데이터에 대한 meta 정보를 생성하여 이 정보를 검증서버에 전송함으로써 검증서버에 데이

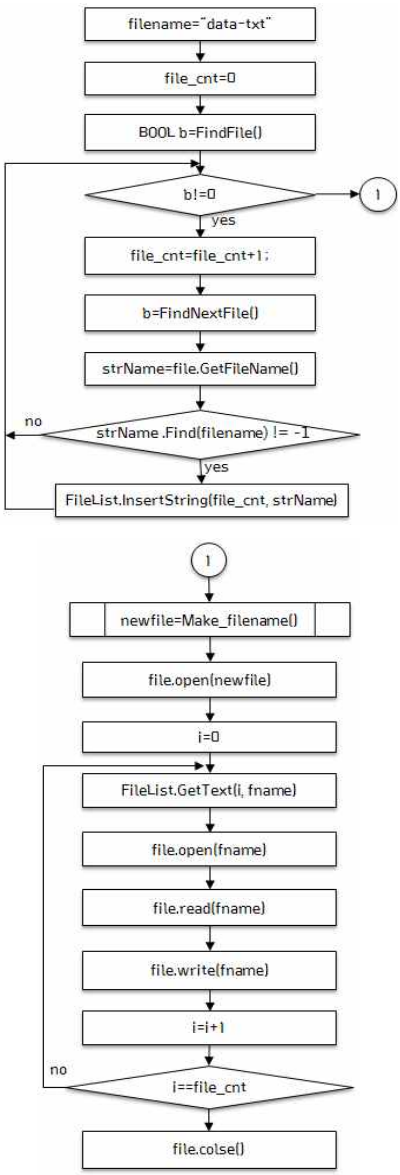


그림 10. 데이터 병합 처리 흐름도

터 무결성 검증을 위임하게 된다.

물론, 클라이언트는 파일 고유 ID, 분할 파일 고유 ID, 암호화된 데이터, meta 데이터, 데이터를 저장한 저장서버 ID, 검증서버 ID를 UFAT에 저장함으로써, 차후에 검증서버가 전송한 데이터의 유효성 결과가 어떤 데이터에 대한 유효성 검증 결과인지를 확인할 수 있도록 하였다.

그림 11은 클라이언트와 검증서버 간의 meta 정보 전송 과정을 설명한 것이고, 단계별 절차는 다음과 같다.

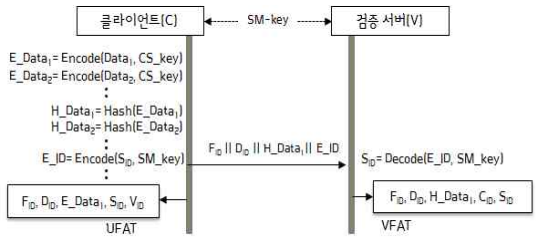


그림 11. meta 정보 처리 과정

[1 단계] 클라이언트는 자신의 비밀키로 암호화한 데이터로 해시하여 검증 노드에 전송할 meta 정보를 생성한다. 그리고 파일 고유 ID, 분할 파일 고유 ID, meta 정보에 대한 저장서버 ID, 데이터의 무결성을 검증할 검증서버 ID를 UFAT에 저장한다.

```
MetaGen(E_data, SID, VID){
  mData = hash(Edata);
  Save(UFAT, FID, DID, E_data, SID, VID);
  return(mData);
}
```

[2 단계] 파일 고유 ID, 분할 파일 고유 ID, meta 정보, 그리고 소지자 노드의 ID를 연접해 검증 노드의 공개키로 암호화시켜 검증 노드에 전달한다.

```
Delegation(mData, SM_key){
  E_ID = Encode(SID, SM_key);
  metaInfo = FID||DID||mData||E_ID||CID;
  return(metaInfo);
}
```

[3 단계] 검증서버는 SM_key로 복호화한 후, 파일 고유 ID, 분할 파일 고유 ID, meta-정보, 데이터 소유자 ID, 그리고 저장서버 ID를 검증 서버의 VFAT에 저장한다. 차후에, 검증서버는 VFAT를 참조하여 저장 서버에 데이터의 무결성 검증 요청 메시지를 전송한다.

```
SaveVFAT(metaInfo){
  SID = Decode(E_ID, SM_key);
  Save(VFAT, FID, DID, mData, CID, SID);
}
```

4. 성능분석

본 논문의 DDMPF는 데이터 분할, 분산 저장, 제3자 검증기법을 적용하여 클라우드 컴퓨팅 환경의 저장 서비스의 보안을 강화하였다. 시뮬레이션 환경은 운영체제 Windows XP Home Edition, CPU 2.20GH, RAM 2GB, 프로그램 언어 Visual C++을 이용하였다.

4.1 보안기법 평가

4.1.1 상호 인증

본 논문의 DDMPF는 EC-DH 알고리즘을 이용해 클라이언트, 저장서버, 검증 서버의 공개키를 생성하고, 상대방의 공개키에 자신의 비밀키로 곱셈을 하여 시스템 마스터키인 SM-key를 생성한다[10]. 이 SM- key로 서로의 신분을 확인하도록 하여 악의적인 사용자가 저장 서버에 임의로 데이터를 저장하는 것을 막고, SM-key는 공개키 암호 알고리즘인 RSA의 1024 bit보다 작은 최대 160bit를 사용하여 적은 데이터로 보안을 강화시켰다.

4.1.2 플로딩 공격 탐지

본 논문의 DDMPF는 검증 서버가 저장 서버에 저장된 데이터의 무결성을 검증하기 위하여 검증요청메시지를 전송하는데, 악의적인 사용자가 이 메시지를 저장 서버에 반복적으로 전송함으로써 플로딩 공격을 할 수 있다. DDMPF는 검증 서버가 검증요청 메시지의 생성 시간을 저장서버에 전송하여 저장서버가 이 생성시간을 확인함으로써 빈번한 검증요청 메시지에 따른 내부 플로딩 공격을 탐지하고, 또한 SFAT에 기록되어 있는 검증 서버의 검증 요청 메시지만을 처리함으로써 외부 플로딩 공격을 탐지한다.

4.1.3 재전송 공격 탐지

검증 서버는 저장 서버의 응답 메시지로 저장 서버의 데이터의 변조 여부를 검증 하는데, 저장 서버는 이 응답 메시지로 재전송 공격을 할 수 있다. 본 논문의 DDMPF에서는 검증 서버가 저장 서버에 요청 메시지를 전송할 때마다 랜덤하게 생성된 새로운 nonce값을 전송하고, 저장 서버는 전달받은 nonce 값과 저장 서버의 ID를 연접해 해시한 값을 재전송한다. 이때 검증서버는 자신이 계산한 해시 값과 재전송된 해시 값을 비교하여 재전송 공격을 탐지하고, 요청 메시지를 작성할 때 nonce 값을 다르게 생성하여 재전송공격을 탐지한다.

4.2 분산 데이터 관리

4.2.1 데이터 중복의 최소화

기존 임계치 기반 비밀분산 방식은 데이터의 중복이 존재하여 비밀분산의 총 크기가 비밀 조각의 개수만큼 증가하는 문제점이 존재한다[14,15]. 본 논문의 DDMPF는 클라이언트가 암호화시켜 분산 저장하는 데이터의 총 크기는 원본 데이터의 크기와 동일하고, 클라이언트가 분할한 데이터만을 암호화하여 분산 저장하므로 저장서버에 저장된 데이터가 중복되지 않으므로 효율적인 처리가 가능하다.

4.2.2 분할파일 크기의 가변성

기존의 임계치 기반 비밀 분산 알고리즘은 비밀 조각의 개수가 비밀 분산 단계에서 결정되어지기 때문에, 한번 비밀 조각으로 분할한 된 비밀 조각의 개수는 재조정이 불가능하다[14,15]. 그러나 이 DDMPF는 분할 파일의 병합으로 분할 파일이 생성된 이후에도 분할 파일의 개수를 자유롭게 조절할 수 있기 때문에 분할 파일의 취급을 용이하다.

4.2.3 파일 분할 및 병합 처리 시간

DDMPF는 파일의 분할과 병합 과정에서 블록 단위의 제한을 받지 않고 기억장치로 연산이 가능한 전체 길이를 한 번에 처리할 수 있다는 장점이 있다.

표 1은 파일 크기가 1KB, 1024KB, 1024MB인 파일을 파일 분할 개수 5일 때, 파일 분할과 병합 처리 시간에 대한 실험 결과를 설명한 것으로 파일의 크기가 커질 수로 처리시간의 조금씩 상승하지만, 사용자가 체감하는 처리 속도에는 큰 차이가 없을 것이다.

표 1. DDMPF 파일의 분할 및 병합 처리시간

파일크기 / 종류	1KB	1024KB	1024MB
파일분할	0.016	0.047	0.063
파일병합	0.015	0.032	0.047

5. 결 론

본 논문은 클라우딩 환경의 보안 문제점들을 해결하고 데이터의 안전성과 신뢰성 향상을 위하여 분할, 분산 저장, 위임 검증 기법을 적용한 DDMPF(Distributed Data Management Protocol using FAT)를 설계하였다.

DDMPF는 첫째, 시스템 마스터키인 SM_key를 생성하고, SM_key로 서로의 신분을 확인하도록 하여 조직 구성요소간의 보안을 강화시켰다. 둘째, 데이터 소유자인 클라이언트가 데이터를 분할하여 자신의 암호키로 데이터를 암호화하여 저장 서버에 분산시켜 저장하므로, 저장서버가 임의로 클라이언트의 데이터를 위임하더라도 클라이언트의 암호화키로 암호화되어 있고, 완전한 데이터가 아니라 분할된 데이터이므로 데이터의 노출 및 임의 양도를 방지할 수 있다. 셋째, 검증서버는 저장서버가 클라이언트의 데이터를 소유하고 있는지 주기적으로 확인하여 그 결과를 클라이언트에게 통보함으로써 데이터의 신뢰성을 향상시켰다. 넷째, 검증단계에서 발생할 수 있는 플로딩 공격은 요청 메시지 생성시간인 TS을 이용하여 탐지하고, 재전송 공격은 검증을 요청할 때마다 생성된 nonce 값을 이용하여 자동적으로 재전송 공격을 탐지하도록 하여, 자가 조직 저장매체의 보안을 강화시켰다. 마지막으로, 클라이언트의 UFAT, 저장 서버의 SFAT, 검증 서버의 VFAT을 두어 각 테이블에 클라이언트, 저장 서버, 검증 서버에 대한 정보를 저장하여 데이터 관리의 효율성을 향상시켰다.

참 고 문 헌

[1] 권혁찬, 문용혁, 구자법, 고선기, 나재훈, 장종수, “P2P 표준화 및 기술동향,” 전자통신동향분석, 제22권, 제1호, pp.11-23, 2007.

[2] 문용혁, 권혁찬, 나재훈, 장종수, “P2P 사용자 인증과 OTP 분석,” 정보보호논문지, 제17권, 제3호, pp. 32-40, 2007.

[3] 김진형, 김윤정, 박춘식, “클라우드 컴퓨팅에서 신뢰하지 않는 서버 데이터의 안전한 접근,” 정보과학회지, 제28권, 제12호 pp. 67-74, 2010.

[4] 임철수, “클라우드 컴퓨팅 보안 기술,” 정보보호학회지, 제19권, 제3호, pp. 14-17, 2009.

[5] 은성경, 조남수, 김영호, 최대선, “클라우드 컴퓨팅 보안 기술,” 전자통신동향분석, 제24권, 제4호, pp. 79-88, 2009.

[6] 강주성, 조성훈, 이옥연, 홍도원, “연관규칙 마이닝에서 랜덤화를 이용한 프라이버시 보호 기법에 관한 연구,” 정보처리학회논문지 C, 제14-C권, 제5호, pp. 439-452, 2007.

[7] NIST, *Special Publication 800-56A, Recommendation for Pair-Wise Key Establishment Schemes using Discrete Logarithm Cryptography*, 2006.

[8] Certicom Research, *Standards for efficient cryptography,* SEC 1: Elliptic Curve Cryptography, Version 1.0, 2000.

[9] 이완복, 노창현, 류대현, “공개키 연산기의 효율적인 통합 설계를 위한 임계경로분석,” 멀티미디어학회논문지, 제8권, 제1호, pp. 79-87, 2005.

[10] 이병관, 정은희, “인프라 클라우딩(Infra Clouding)환경에서 자가조직 저장매채의 보안을 위한 3자간 협상 프로토콜 설계,” 멀티미디어학회논문지, 제14권 제10호, pp. 1303-1310, 2011

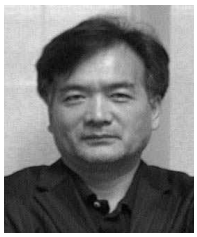
[11] N. Koblitz, “Elliptic Curve Cryptosystems,” *Mathematics of Computation*, Vol. 48, No. 177, pp. 203-209, 1987.

[12] V. Miller, “Uses of Elliptic Curves in Cryptography,” *Advances in Cryptology, Proc. of Crypto'85, LNCS(218)*, pp. 417-426, 1986.

[13] Y. Desmedt, “Society and Group Oriented Cryptography: A New Concept,” *Advances in Cryptology-Proceeding of Crypto'87 (Lecture Notes in computer Science 293)*, pp. 120-127, 1988.

[14] A. Sharmir, “How to Share a Secret,” *Communication ACM*, Vol. 22, Issue 11, pp. 612-613, 1979.

[15] 박남제, 송유진, “스마트 그리드 환경에서 ANOT 암호화 방법을 이용한 안전한 분산 데이터 관리 구조 방안,” 한국통신학회논문, 제35권, 제10호, pp. 1458-1470, 2010.



이 병 관

1979년 2월 부산대학교 기계설계학과 공학사
1986년 2월 중앙대학교 전자계산공학과 공학석사
1990년 2월 중앙대학교 전자계산학과 공학박사

1988년 3월～현재 관동대학교 컴퓨터학과 교수
관심분야 : 네트워크 보안, 컴퓨터 네트워크, 전자상거래

정 은 희

1991년 2월 강릉대학교 통계학과 이학사
1998년 2월 관동대학교 전자계산공학과 공학석사
2003년 2월 관동대학교 전자계산공학과 공학박사

2003년 9월～현재 강원대학교 지역경제학과 부교수
관심분야 : 네트워크 보안, 웹 프로그래밍, 전자상거래

양 승 해

2000년 2월 관동대학교 컴퓨터공학과 공학학사
2002년 2월 관동대학교 전자계산공학과 공학석사
2005년 8월 관동대학교 전자계산공학과 공학박사

2007년 12월～현재 (주)CDS 연구실장
관심분야 : 네트워크 보안, 전자상거래