Journal of The Korea Institute of Information Security & Cryptology ISSN 1598-3986(Print)
VOL.25, NO.1, Feb. 2015 ISSN 2288-2715(Online)
http://dx.doi.org/10.13089/JKIISC.2015.25.1.155

분산 스토리지 시스템에서 데이터 중복제거를 위한 정보분산 알고리즘 및 소유권 증명 기법

신 영 주†‡
한국전자통신연구원 부설연구소

Information Dispersal Algorithm and Proof of Ownership for Data Deduplication in Dispersed Storage Systems

Youngjoo Shin†‡
The Attached Institute of ETRI

요 약

저장된 데이터에 대한 높은 가용성과 기밀성을 보장하는 정보분산 알고리즘은 클라우드 스토리지 등 장애 발생 비율이 높고 신뢰할 수 없는 분산 스토리지 시스템에서 유용한 방법이다. 스토리지에 저장되는 데이터의 양이 증가하면서 IT 자원을 효율적으로 활용하기 위한 데이터 중복제거기법이 많은 주목을 받고 있으며 이에 따라 데이터 중복제거가 가능한 정보분산기법에 대한 연구도 필요한 시점이다. 본 논문은 분산 스토리지 시스템에서 클라이언트 기반 중복 제거를 위한 정보분산 알고리즘과 소유권 증명 기법을 제안한다. 제안하는 방법은 저장공간 뿐만 아니라 네트워크 대역 절감이 가능하여 높은 효율성을 얻을 수 있으며 신뢰할 수 없는 스토리지 서버와 악의적인 클라이언트로부터 안전성을 보장할 수 있다.

ABSTRACT

Information dispersal algorithm guarantees high availability and confidentiality for data and is one of the useful solutions for faulty and untrusted dispersed storage systems such as cloud storages. As the amount of data stored in storage systems increases, data deduplication which allows to save IT resources is now being considered as the most promising technology. Hence, it is necessary to study on an information dispersal algorithm that supports data deduplication. In this paper, we propose an information dispersal algorithm and proof of ownership for client-side data deduplication in the dispersed storage systems. The proposed solutions allow to save the network bandwidth as well as the storage space while giving robust security guarantee against untrusted storage servers and malicious clients.

Keywords: Information Dispersal Algorithm, Proof of Ownership, Data Deduplication

I. 서 론

정보분산 알고리즘은 하나의 데이터를 여러 개의 조각으로 분할 한 뒤 각 조각을 물리적, 지역적으로 떨어진 개별 스토리지에 분산 저장하는 방법이다. 스토리지에 장애가 발생하였을 때 데이터 접근을 보장하는 가용성을 제공할 뿐 아니라 데이터를 나누어 분산하므로 데이터에 대한 기밀성도 제공하는 특징을 지닌다. 따라서 장애 발생 비율이 높고 신뢰할 수 없는 분산 스토리지 환경에서 유용하게 쓰일 수 있다. 최근에

접수일(2014년 11월 21일), 수정일(2015년 1월 19일), 게재확정일(2015년 1월 19일)
† 주저자, s.youngjoo@gmail.com
‡ 교신저자, s.youngjoo@gmail.com(Corresponding author)

는 여러 개의 클라우드를 묶어 하나의 스토리지로 사용하는 Cloud-of-Clouds 개념의 정보분산 알고리즘에 대해서도 많은 연구가 이루어지고 있다 [1-3].

한편, 스토리지에 저장하는 데이터의 양이 증가하면서 비용 절감의 필요성이 강하게 대두됨에 따라 데이터 중복 제거 기술이 많은 주목을 받고 있다[4]. 데이터 중복 제거란 동일한 내용의 데이터가 있을 경우 하나만 남겨두고 나머지는 논리적 링크로 대체하는 기술을 말한다. 중복 제거 기술은 중복이 탐지되는 위치에 따라 서버(스토리지) 기반과 클라이언트 기반 중복 제거로 분류할 수 있다[5-7]. 클라이언트에서 중복을 탐지하는 경우 데이터 업로드를 생략할 수 있으므로 스토리지 저장공간 뿐만 아니라 데이터를 전송하는데 필요한 네트워크 대역폭도 절감할 수 있어 서버 기반 방식에 비해 높은 효율성을 얻을 수 있다.

최근에 데이터 중복 제거가 가능한 정보분산 기법이 제안되기도 하였으나[3] 클라이언트 기반 중복 제거 방식을 고려하여 설계되지는 않았다. 분산 스토리지 환경에서 클라이언트 기반 중복 제거를 적용하기 위해서는 두 가지 사항을 고려하여야 한다. 첫 째, 데이터를 소유하고 있지 않은 클라이언트가 스토리지 서버를 속이지 못하도록 하는 소유권 증명 프로토콜(Proof of ownership) 이 필요하다[6]. 소유권 증명 프로토콜은 데이터 F 를 가지고 있는 스토리지 서버와 클라이언트가 challenge 와 response 를 주고받으면서 F 의 내용을 증명하고 이를 검증한다. 프로토콜의 안정성(Soundness)을 보장하기 위해서는 F 의 내용 중 공격자가 알지 못하는 미지의 정보가 전체로 확장이 되도록 데이터가 인코딩 되어야 하며 증명 생성 과정에서 해쉬 값 등 상대적으로 크기가 작은 상태 정보가 이용되지 않아야 한다[6]. 둘째, 분산 스토리지 환경은 별도의 중앙 서버가 존재하지 않으므로 각각의 스토리지 서버가 F 의 분산된 조각을 가지고 클라이언트와 소유권 증명 프로토콜을 개별적으로 실행하여 원래 데이터의 소유권을 검증할 수 있는 방법이 필요하다.

본 논문에서는 분산 스토리지 환경에서 클라이언트 기반 데이터 중복 제거가 가능한 CMC-RS 정보분산 알고리즘과 분산 소유권 증명 방법을 제안한다. CMC-RS 는 CMC (CBC-mask-CBC) 암호 알고리즘[8]과 Reed-Solomon 부호[9]를 결합한 정보분산 알고리즘으로 스토리지 서버가 안전하고 신뢰할 수 있는 방법으로 클라이언트의 데이터 소유 여부를 검증할 수 있는 것을 보장한다. 분산 소유권 증명 방법은 개별 스토리지 서버가 독립적으로 분산된 데이터 조각을 가지고 원래 데이터의 소유권을 검증하는 방법이다. 본 제안 방법은 전체 스토리지 서버 중 일부의 서버들과 프로토콜을 실행하여도 데이터의 소유권 증명 및 검증이 가능한 장점을 제공한다.

본 논문의 구성은 다음과 같다. 2장에서 본 연구와 관련된 기존 연구 내용을 기술하고 3장에서 설계 목표 및 제안 방식의 개괄적인 내용을 기술한다. 그리고 4장과 5장에서 CMC-RS 정보분산 알고리즘과 분산 소유권 증명 프로토콜을 각각 제시한다. 6장에서 제안한 방법에 대한 안전성 및 성능을 분석하고 7장에서 결론으로 끝을 맺는다.

II. 관련 연구

(n,k)-정보분산 알고리즘$(n>k>0)$은 데이터 F 를 n 개의 조각으로 나누어 스토리지 서버에 분산하는 알고리즘으로 다음과 같은 특성을 갖는다.

- 가용성 : n 개의 스토리지 서버 중 최대 $n-k$ 개의 서버에 장애가 발생하여도 나머지 k 개의 조각으로부터 원래 데이터 F 를 복구 할 수 있다.
- 기밀성 : $k' < k$ 개의 서버가 공모하여 k' 개의 조각을 가졌을 때 이로부터 원래 데이터 F 에 대한 정보를 완전히 얻는 것은 매우 어렵다.

Sharmir등의 비밀 분산 알고리즘 (SSSS) [10]은 F 를 지나는 $k-1$차 다항식을 무작위로 선택한 다음 다항식 위의 n 개의 값을 생성하여 스토리지에 분산한다. SSSS 는 $k-1$개 이하의 조각들로부터 F 에 대한 어떤 정보도 얻을 수 없는 높은 수준의 기밀성을 보장할 수 있으나 저장 공간이 n 배의 크기로 증가하므로 비효율적이다. 램프 비밀 분산 알고리즘 (RSSS)[11]은 SSSS의 개선된 알고리즘으로 r 개 $(0 \le r < k)$ 이하의 조각 획득에 대해서만 기밀성을 보장하는 대신 크기는 $n/(k-r)$ 배 증가하여 저장 효율성을 높였다.

Rabin 등의 정보분산 알고리즘 (IDA)[12]은 크기 증가율이 n/k 로 높은 저장 공간 효율성을 제공하나 1개의 조각으로부터 원래 데이터의 부분 정보가 노출 될 수 있어 매우 낮은 수준의 기밀성을 갖는다. Resch 등은 Rabin 의 IDA를 개선하여 저장효율성은 유지하면서 높은 기밀성을 갖는 AONT-RS라는 정보분산 알고리즘을 제안하였다[13]. AONT-RS

는 데이터 F 를 AONT (All-or-nothing Transform)[14] 기법으로 k 개의 조각으로 분할한 뒤 이 조각을 다시 RS (Reed-Solomon) 부호를 수행하여 최종적으로 n 개의 조각으로 분산한다.

한편 정보분산 알고리즘은 기본적으로 랜덤 알고리즘이다. 따라서 알고리즘에 의해 분산된 조각들은 난수화된 형태로 저장된다. 이는 스토리지 서버에서 동일한 데이터의 중복 여부를 판단 할 수 없게 만든다. Li 등은 RSSS[11]와 AONT-RS[13]를 개선하여 서버 측에서 데이터 중복 제거가 가능한 정보분산 알고리즘 CRSSS 와 CAONT-RS 를 제안하였다[3]. 기본 아이디어는 수렴 암호화 방식 (Convergent encryption)[15] 과 동일하며 데이터의 해쉬 값 등으로부터 계산된 값을 난수 값 대신 사용한다.

CRSSS 알고리즘은 다음과 같이 동작한다. 우선 데이터 F 를 $k-r$ 개의 워드로 구성된 l개의 블록 $F_1,...,F_l$ 으로 분할한 다음 각각의 블록 F_j $(1 \le j \le l)$ 에 대해 r 개의 서로 다른 해쉬 알고리즘 h_i $(1 \le i \le r)$ 으로 해쉬 값 $h_i(F_j)$ 을 계산한다. $k-r$ 개의 워드와 r 개의 해쉬 값으로 구성된 총 k 개의 워드에 대해 Rabin 의 IDA 알고리즘을 적용하여 n 개의 코드워드를 생성하고 코드워드를 스토리지 서버에 분산한다.

Li 등이 제안한 두 번째 알고리즘인 CAONT-RS 는 AONT-RS 알고리즘[13]과 유사하다. F 가 s 개의 워드 $d_1,...,d_s$ 로 이루어져있다고 하자. 이로부터 $s+1$ 개의 워드 $c_1,...,c_{s+1}$ 을 다음과 같이 계산하여 얻는다.

$$c_i = d_i \oplus E_K(i) \ , 1 \le i \le s \quad (1)$$

$$c_{s+1} = K \oplus h(c_1 \| ... \| c_s) \quad (2)$$

여기서 E_K 와 h는 각각 블록암호와 해쉬 알고리즘이고 K 는 F 의 해쉬 값으로부터 계산된 암호키이다. 워드 $c_1,...,c_{s+1}$ 은 다시 k 개의 동일한 크기의 조각으로 나눈 뒤 이 조각들에 대해 RS 부호를 수행하여 n 개의 조각을 생성한다.

CRSSS 와 CAONT-RS 는 기본적으로 서버 기반 데이터 중복 제거를 고려하여 설계되었다. 클라이언트 기반의 방식을 사용하기 위해서는 소유권 증명 프로토콜이 구현 가능해야한다. 그러나 CRSSS 또는 CAONT-RS 가 적용된 분산 스토리지 시스템에서는 높은 안전성을 갖는 프로토콜의 구현이 어렵다.

클라이언트 기반 중복제거를 적용한 분산 스토리지 시스템이 있고 데이터 F 의 일부 ($i.e.$, l 개의 블록 중 첫 번째 블록 F_1)를 알지 못하는 공격자를 가정해 보자. CRSSS를 사용하는 시스템이라면 각 스토리지 서버는 l 개의 코드워드로 이루어진 조각들이 분산 저장되어 있을 것이다. 공격자는 미지의 블록 F_1을 제외한 나머지 $l-1$ 개의 블록에 대해서는 그 정보를 전부 가지고 있으므로 공격자가 생성한 CRSSS 조각의 대부분 ($i.e.$, $1-1/l$ 의 비율)은 스토리지에 저장된 것과 동일하다. 따라서 서버에게 높은 확률로 F의 내용을 증명할 수 있게 된다.

CAONT-RS 알고리즘도 동일한 문제를 가지고 있다. 이 알고리즘은 AONT 암호화 과정에서 F 의 해쉬 값을 암호 연산의 암호키 K 로 사용한다. 따라서 F 의 일부 정보를 모르는 공격자는 올바른 K 를 계산할 수 없으므로 동일한 조각을 생성할 수 없다. 그러나 Halevi 등이 제시한 공격 모델[6]을 고려한다면 올바른 암호키 K 를 포함하여 제한적으로 필요한 정보를 공격자에게 제공할 수 있는 조력자 (Accomplice) 를 생각할 수 있다. 공격자는 조력자의 도움을 받아 미지의 부분을 제외한 조각의 나머지 전체 부분도 동일하게 생성해낼 수 있게 된다.

요약하자면, CRSSS 와 CAONT-RS 의 문제는 F 에서 공격자가 알지 못하는 미지의 부분이 생성된 조각의 전체로 확대되지 않는다는 것이다. 이는 소유권 증명 프로토콜의 안정성을 보장하기 어렵게 하며 공격자는 높은 확률로 프로토콜을 통과하여 스토리지 서버를 속일 수 있게 된다.

III. 설계 목표 및 제안하는 방식의 개요

3.1 설계 목표

3.1.1 저장 데이터에 대한 기밀성

클라이언트 입장에서 스토리지 서버는 신뢰할 수 없다. 따라서 스토리지 서버로부터 저장 데이터에 대한 기밀성이 반드시 요구된다.

본 논문에서는 스토리지 서버는 클라이언트의 저장 데이터에 대한 사전 정보를 거의 가지고 있지 않으며 honest-but-curious 하다고 가정한다. 즉, 스토리지 서버는 저장 데이터를 훼손하지 않고 프로토콜을 올바르게 실행하지만 클라이언트 소유의 데이터에 대

한 정보를 얻으려 노력한다. 이를 위해 충분히 많은 컴퓨팅 자원을 활용할 수 있으며 n 개의 스토리지 서버 중 최대 $k-1$ 개의 서버가 공모하여 정보를 얻기 위한 공격을 실행할 수 있다.

3.1.2 소유권 증명의 안정성

소유권 증명 프로토콜에 대한 안정성은 Halevi 가 정의한 모델[6]을 따른다. 증명하려는 데이터 F 에서 τ 비트 크기만큼의 일부 정보를 알지 못하는 공격자를 가정하자. 즉, 이 공격자에게는 최소 엔트로피 (Min entropy) 가 τ 인 F 의 임의의 확률 분포가 주어졌다고 생각 할 수 있다. 이 공격자에게 최대 $\tau-\sigma$ 비트의 정보를 제공하는 조력자가 존재한다고 가정한다. 공격자는 조력자의 도움을 받아 적어도 σ 비트 크기의 정보를 갖지 못한 채 t $(0<t\leq n)$ 대의 스토리지 서버와 소유권 증명 프로토콜을 실행한다. 이 공격자가 서버를 속이고 F 에 대한 증명에 성공하여 소유권을 획득할 확률은 매우 작아야한다.

3.2 제안하는 방법의 개요

분산 스토리지 환경에서 클라이언트 기반 중복 제거를 위한 CMC-RS 정보분산 알고리즘과 분산 소유권 증명 기법에 대해 그 개요를 간략히 기술한다.

3.2.1 CMC-RS 정보분산 알고리즘

CMC-RS 정보분산 알고리즘은 CMC 암호 알고리즘과 RS 부호가 결합된 구조를 가지고 있다. CMC 암호 알고리즘은 디스크 암호화 등의 목적을 위해 설계된 블록 암호화 알고리즘의 동작 모드로 [8] 동일한 암호키와 평문에 대해 랜덤한 암호문이 생성되는 성질을 지닌 tweakable 알고리즘으로 설계되었다. 본 논문에서는 고정된 tweak으로 CMC 암호 알고리즘 (Table 1., Table 2.)을 사용한다.

CMC 암호 알고리즘의 특징은 가변길이 블록에 대해서 강한 의사난수 치환 (Strong pseudorandom permutation) 이라는 것이다 [8]. 이는 CMC 암호 알고리즘과 난수 치환 (Random permutation)이 계산적으로 구분 불가능함을 의미한다. 따라서 CMC 암호 알고리즘으로 생성된 암호문은 높은 기밀성을 제공할 뿐 아니라 평문의 1 비트가 변경이 되어도 암호문 전체로 변경이 확산되는 특성을 갖는다. 이러한 특성은 소유권 증명 프로토콜의 데이터 인코딩 방법으로 적합하다. CMC 암호화 알고리즘으로 암호화된 데이터는 RS 부호로 여러 개의 조각으로 분산된다.

3.2.2 분산 소유권 증명 기법

본 논문에서 제안하는 분산 소유권 증명 기법은 분산 스토리지 환경의 각 스토리지 서버가 독립적으로 클라이언트와 프로토콜을 실행하여 데이터의 소유권을 검증할 수 있게 해준다. 이를 위해 Merkle 트리 증명 프로토콜[6] 과 임계 서명 (Threshold signature) 기법[16,17]을 이용한다.

구체적으로, 클라이언트와 스토리지 서버는 CMC-RS 알고리즘에 의해 분산된 F 의 조각에 대해서 Merkle 트리 증명 프로토콜[6]을 실행한다. 스토리지 서버는 검증을 통과한 클라이언트에게 조각에 대한 소유 정보 tag_F 를 생성하고 분배된 서명키로 서명하여 전송한다. n 개의 스토리지 서버 중 t 개 이상의 서버로부터 서명된 tag_F 를 얻게 되면 클라이언트는 tag_F 의 온전한 서명값을 획득 할 수 있게 된다. 클라이언트는 이후에 임의의 스토리지 서버에 tag_F 를 제시하여 서명 검증을 받고 조각을 모아 원래 데이터 F 를 복구 할 수 있다.

임계 서명을 이용한 분산 소유권 증명 기법은 전체 스토리지 중 일부와 프로토콜을 실행하여도 소유권 증명이 가능한 장점을 제공한다.

IV. CMC-RS 정보분산 알고리즘

4.1 데이터 F 의 정보분산 과정

F에 대한 CMC-RS 정보분산 알고리즘의 실행 과정을 3단계로 나누어 기술한다. (Fig.1.)

우선 F 에 대해 CMC 암호화를 한다. F 를 SHA-256[18] 등의 해쉬 알고리즘으로 계산한 해쉬

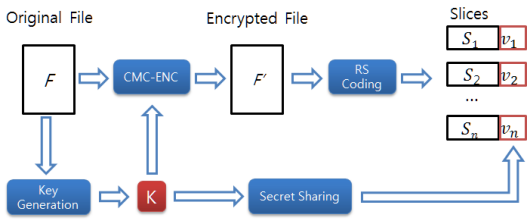

Fig. 1. The overview of CMC-RS

값으로부터 유도된 값을 암호키 K 로 사용한다. 암호화 알고리즘 **CMC-ENC**$_K$ 은 Table 1.에 기술되어 있다. (블록암호 알고리즘 E_K 로는 AES를 사용한다.) F를 m개의 128비트 크기의 블록 $P_1,...,P_m$으로 분할한 뒤 **CMC-ENC**$_K$ 을 실행하여 암호화된 데이터 F'를 생성한다.

다음으로 암호화된 데이터 F'에 대해 RS 부호 연산을 수행한다. F' 는 유한체 $GF(2^w)$ 의 심볼들로 이루어져있고 전체 심볼의 개수는 N 개라 하자. F'를 크기가 동일한 k 개의 부분 데이터 $F_1',...,F_k'$ 으로 분할하면 하나의 부분 F_i' $(1 \le i \le k)$에는 N/k 개의 심볼 $b_{i,1},...,b_{i,N/k}$ 을 갖게 된다. 각 부분에서 하나씩 가져온 심볼 $b_{1,j},...,b_{k,j}$ $(1 \le j \le N/k)$ 에 대해 RS 부호 연산을 수행하여 n 개의 심볼 $c_{1,j},...,c_{n,j}$ 를 생성한다. 그리고 이를 n 개의 조각에 각각 추가한다. (Fig.2.) RS 부호는 systematic erasure 부호이므로 $c_{i,j} = b_{i,j}$ $(1 \le i \le k)$이다. 즉, 생성된 n개의 심볼에는 k개의 F'의 심볼이 포함된다. 따라서 원래 데이터 F'의 심볼은 두고 나머지 $n-k$개의 심볼들만 생성하면 되므로 효율적인 계산이 가능하다. 이 과정이 완료되면 최종적으로 N/k 개의 심볼로 이루어진 n 개의 조각 $S_1,...,S_n$ 가 생성된다.

마지막으로, CMC 암호화에 사용된 암호키 K 를 (n,k)-비밀분산 기법[10]을 이용하여 n 개의 분산값 $v_1,...,v_n$ 으로 분할한다.

생성된 조각 S_i $(1 \le i \le n)$ 와 분산값 v_i 는 i 번째 스토리지 서버에 전송한다.

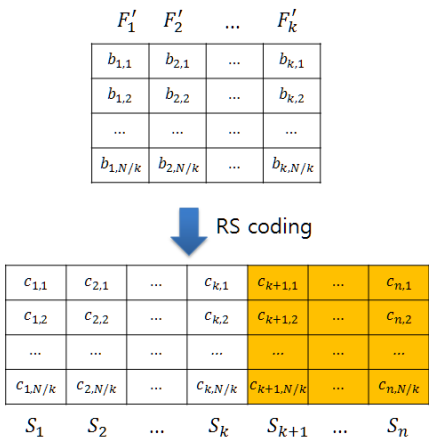

Fig. 2. An example of RS coding with F'

Table 1. CMC encryption algorithm with a fixed tweak [8]

CMC-ENC$_K$ $(P_1||...||P_m)$
$PPP_0 \leftarrow 0$
for $i \leftarrow 1$ **to** m
$PP_i \leftarrow P_i \oplus PPP_{i-1}$
$PPP_i \leftarrow E_K(PP_i)$
$M \leftarrow 2 \times (PPP_1 \oplus PPP_m)$
for $i \leftarrow 1$ **to** m
$CCC_i \leftarrow PPP_{m+1-i} \oplus M$
$CCC_0 \leftarrow 0$
for $i \leftarrow 1$ **to** m
$CC_i \leftarrow E_K(CCC_i)$
$C_i \leftarrow CC_i \oplus CCC_{i-1}$
return $C_1||...||C_m$

4.2 여러 스토리지에 분산된 데이터 F의 복구

CMC-RS 정보분산 알고리즘에 의해 n 개의 스토리지 서버에 분산되어 저장된 데이터 F 는 다시 원래대로 복구할 수 있다. n 개의 스토리지 서버 중 k 개의 서버로부터 저장된 조각 S_i 와 암호키 분산값 v_i $(1 \le i \le n)$을 가져온다. 조각들로부터 암호화된 데이터 F'를 복구하고 분산값으로부터 암호키 K를 복구한다. F' 는 K 와 복호화 알고리즘 **CMC-DEC**$_K$ (Table 2.)을 사용하여 원래의 데이터 F 로 복구한다.

Table 2. CMC decryption algorithm with a fixed tweak [8]

CMC-DEC$_K$ $(C_1||...||C_m)$
$CCC_0 \leftarrow 0$
for $i \leftarrow 1$ **to** m
$CC_i \leftarrow C_i \oplus CCC_{i-1}$
$CCC_i \leftarrow D_K(CC_i)$
$M \leftarrow 2 \times (CCC_1 \oplus CCC_m)$
for $i \leftarrow 1$ **to** m
$PPP_i \leftarrow CCC_{m+1-i} \oplus M$
$PPP_0 \leftarrow 0$
for $i \leftarrow 1$ **to** m
$PP_i \leftarrow D_K(PPP_i)$
$P_i \leftarrow PP_i \oplus PPP_{i-1}$
return $P_1||...||P_m$

4.3 효율성 향상을 위한 다중 CMC 암호화 방법

암호화할 데이터 F 가 m 개의 블록으로 구성되어

있다고 할 때 CMC 암호화 방식은 기본적으로 $2m$ 번의 E_K 블록암호 알고리즘을 수행하게 된다. F 의 크기가 매우 큰 경우 메모리에서 한 번에 처리하는 것은 불가능하며 디스크에서 적어도 2번에 걸쳐 읽고 쓰는 I/O 과정이 불가피하게 발생하게 되므로 성능 저하가 나타날 수 있다.

용량이 큰 데이터의 처리 시 과도한 디스크 I/O 에 의한 전체 성능의 저하를 방지하기 위한 방안으로 데이터 F 를 동일한 크기의 여러 분할 데이터 $F_1,...,F_\delta$ 로 나누어 각각에 대해 다중으로 암호화를 하는 방법을 생각해볼 수 있다. (Fig.3.) 분할의 크기 $|F_i|$ $(1 \le i \le \delta)$를 메모리의 크기보다 작게 설정하면 어떠한 용량의 파일이라도 전체적으로 한 번의 읽기, 쓰기 I/O 만으로 암호화를 할 수 있다.

분할의 크기 $|F_i|$ 를 선택할 때는 한 가지 고려사항이 있다. $|F_i|$ 를 작게 설정한다면 조력자는 해당 크기만큼의 필요한 데이터를 공격자에게 전송하여 공격을 성공하게 만들 수 있다. 조력자가 전송할 수 있는 데이터의 최대 크기를 T 라고 할 때 $|F_i|$ 의 크기는 적어도 T 와 같거나 큰 값을 선택해야한다. Halevi 등은 시스템 사양, 네트워크 대역폭 등을 고려하여 T 를 64MB 로 설정하였다[6]. 최근에는 일반적인 PC 는 4GB 이상의 메모리가 기본 탑재되므로 본 논문에서는 T 를 적어도 512MB 또는 1GB 로 설정할 것을 제안한다.

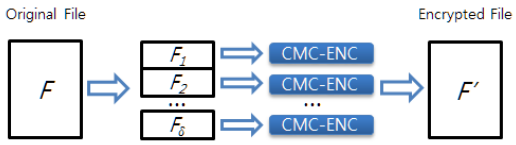

Fig. 3. An example of multi-CMC with F

V. 분산 소유권 증명 기법

n 개의 스토리지 서버에 F 가 조각 $S_1,...,S_n$ 으로 나뉘어 분산 저장되어있다고 하자. 이후 동일한 F에 대해 클라이언트의 업로드 요청이 생길 경우 소유권 증명 프로토콜을 실행하여 파일 전송을 생략할 수 있다. 본 장에서는 n 개의 스토리지 서버 중 t 개의 서버와 독립적으로 소유권 증명 프로토콜을 실행하여 검증하는 방법에 대하여 기술한다.

5.1 배경 지식

5.1.1 Merkle 트리

Merkle 트리 $MT_{h,\beta}(F)$ 는 데이터 F 의 β 비트 크기의 블록들을 리프 노드로 하고 자식 노드 쌍을 해쉬 알고리즘 h로 연산한 값을 중간 노드로 하는 이진 트리이다. Fig.4. 는 8개의 블록 $B_1,...,B_8$ 으로 이루어진 데이터 F 에 대한 $MT_{h,\beta}(F)$ 의 예를 보여준다. 중간 노드 $n_1,...,n_4$ 는 자식 노드의 해쉬 값 $v_1 = h(B_1 \| B_2),...,v_4 = h(B_7 \| B_8)$ 을 가지고 있다. 노드 n_5 는 $v_5 = h(v_1 \| v_2)$ 을 가지고 있으며 n_6 와 루트 노드 n_8 도 v_6 와 v_8 을 각각 계산하여 갖게 된다.

리프 노드 B 의 형제 경로 (Sibling path) 는 B 를 포함하면서 B 에서 루트 노드까지의 경로 상의 각 노드의 형제 값들의 집합이다. 예를 들어 Fig.4. 에서 B_1의 형제 경로는 $P = \{B_1, B_2, v_2, v_6\}$ 이다. 형제 경로 P 의 값들을 이용하여 $MT_{h,\beta}(F)$ 의 루트 노드의 값 v_8을 계산할 수 있다. 만약 계산한 값이 원래 루트 노드의 값과 동일하다면 P 는 유효한 형제 경로라고 한다.

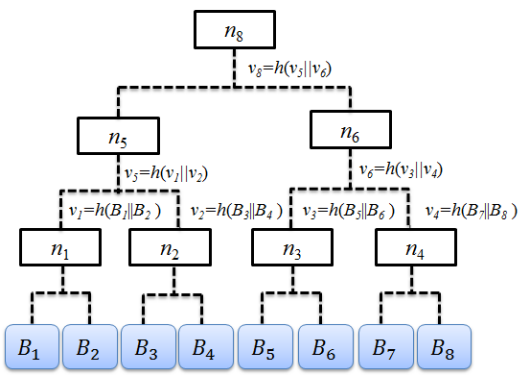

Fig. 4. An example of Merkle tree with 8 blocks

5.1.2 임계 서명

(n,t)-임계 서명은 n 명의 서명자 중 t 명 이상으로부터 서명을 받은 메시지에 대해서만 검증이 가능한 서명 기법이다[16,17]. 최대 $t-1$ 명의 공모 공격에 대해 올바른 서명값을 만들어 낼 수 없으며 서명키 없이 서명을 위조할 수 없는 안전성을 만족한다. 임계 서명은 다음과 같이 4개의 알고리즘으로 구성된다.

- $TKeygen(1^{\lambda})$: 안전성 파라미터 1^{λ}를 입력으로 하여 n명의 서명자가 검증키 vk와 서명키 sk를 생성한 뒤 vk는 공개하고 sk는 n 개의 서명키 sk_i $(1 \le i \le n)$으로 나누어 각 서명자에게 분배한다.
- $TSignature(sk_i, M)$: sk_i 와 메시지 M 을 입력으로 받아 서명 분배값 (Signature share) $\{M\}_{sk_i}$ 을 생성한다.
- $TCombine(\{M\}_{sk_1},...,\{M\}_{sk_t})$: t 개의 서명 분배값 $\{M\}_{sk_1},...,\{M\}_{sk_t}$ 을 입력으로 받아 원래 서명값 $\{M\}_{sk}$ 을 생성한다.
- $TVerify(vk, \{M\}_{sk})$: 검증키 vk 와 서명값 $\{M\}_{sk}$ 을 입력으로 받아 이 서명값이 올바르게 서명되었는지 검증한다.

5.2 Merkle 트리 증명 프로토콜

하나의 스토리지 서버와 클라이언트 간의 소유권 증명은 기본적으로 Halevi 등이 제안한 프로토콜 [6]을 따른다. 스토리지 서버는 조각 S_i 가 업로드 되면 $MT_{h,\beta}(S_i)$ 를 생성하고 루트 노드 값 r_i 과 리프 노드의 개수 l_i 를 보관한다. S_i 에 대해 클라이언트와 서버간의 Merkle 트리 증명 프로토콜은 다음과 같다.

(1) 서버는 무작위로 u개의 인덱스 c_j $(1 \le j \le l_i)$ 를 선택하여 클라이언트에게 전달한다.
(2) 클라이언트는 S_i 로부터 $MT_{h,\beta}(S_i)$ 를 생성한 뒤 각 인덱스 c_j 에 해당하는 리프 노드의 형제 경로 P_j 들을 생성하여 서버에게 전달한다.
(3) 서버는 전달 받은 P_j 가 전부 유효한 형제 경로인지 검증한다. 즉, P_j 로부터 루트 노드의 값을 계산하고 이 값이 보관된 값 r_i 과 일치하는 지 확인한다. 모든 형제 경로 P_j 가 유효하다면 클라이언트에게 'Accept' 메시지를 출력하고 그렇지 않으면 'Fail' 메시지를 출력한다.

5.3 분산 소유권 증명 기법

n 개의 스토리지 서버는 시스템의 초기화 과정에서 $TKeygen(1^{\lambda})$ 알고리즘을 실행하여 서명키 sk_i $(1 \le i \le n)$ 를 나누어 가지고 있다고 가정 하자.

F 를 분산 스토리지 시스템에 업로드 하려는 클라이언트는 우선 F 의 존재 여부를 확인한다. 스토리지에 아직 존재하지 않는다면 CMC-RS 알고리즘으로 조각을 생성한 뒤 업로드 한다. 이미 존재한다면 클라이언트는 다음과 같이 분산 소유권 증명을 실행한다.

(1) F 에 대해 CMC-RS 알고리즘을 실행하여 n 개의 조각 $S_1,...,S_n$ 을 생성한다.
(2) 조각 S_i $(1 \le i \le n)$에 대해 해당 조각을 보유하고 있는 서버와 5.2절의 Merkle 트리 증명 프로토콜을 실행한다. 실행 결과 'Accept' 이면 스토리지 서버는 서명키 sk_i 를 가지고 서명 분배값 $\{tag_F\}_{sk_i} = TSignature(sk_i, tag_F)$ 를 생성하여 이를 클라이언트에게 전달한다.
(3) 클라이언트는 과정(2) 를 적어도 t $(t \le n)$ 개 이상의 스토리지 서버와 실행한다.
(4) 클라이언트는 t 개의 서버와 프로토콜을 실행하여 얻은 서명 분배값을 가지고 $TCombine$ 알고리즘을 실행하여 서명값 $\{tag_F\}_{sk}$ 을 생성한다.

서명값 $\{tag_F\}_{sk}$ 가 정상적인 서명인지의 여부는 $TVerify$ 알고리즘을 이용하여 공개된 검증키 vk 로 검증할 수 있다. 클라이언트는 임의의 스토리지 서버에 $< tag_F, \{tag_F\}_{sk} >$ 를 제시하여 서명검증을 받고 서버에 저장된 조각을 다운로드 받을 수 있다.

본 제안 기법에서 사용하는 임계 서명의 임계값 t 는 $t = k$ 로 제한 한다. 만약 $t < k$ 이면 k개보다 적은 수의 스토리지 서버들이 서로 공모하여 정상적인 $< tag_F, \{tag_F\}_{sk} >$를 생성해낼 수 있기 때문이다.

VI. 안전성 및 성능 분석

6.1 안전성 분석

6.1.1 데이터 기밀성

데이터 F 는 CMC 암호 알고리즘으로 암호화를 거쳐 각 스토리지 서버에 분산되므로 스토리지 서버는 하나의 조각 S_i $(1 \le i \le n)$ 으로부터 F 에 대한 정보를 계산할 수는 없다.

최대 $k-1$개의 서버가 공모하여 정보를 알아내려는 공격을 생각해보자. $k-1$개의 조각으로부터 암호

화된 데이터 F'의 일부 정보는 얻을 수 있으나 암호화되어있으므로 복호화하기 위한 암호키 K가 필요하다. K는 (n,k)-비밀분산 기법으로 분산되었으므로 마찬가지로 $k-1$ 개의 분산 값들을 가지고 K에 대한 어떠한 정보도 얻을 수 없다.

6.1.2 소유권 증명의 안정성

F에 대해 최소 엔트로피 τ를 가지고 있는 공격자가 있고 이 공격자는 조력자로부터 암호키 K를 포함하여 최대 $\tau-\sigma$ 비트만큼의 F에 대한 정보를 획득하였다고 하자. 이 공격자에 대해 본 논문에서 제안한 분산 소유권 증명 기법의 안정성을 분석한다.

F가 CMC-RS 정보분산 알고리즘에 의해 n개의 조각 $S_1,...,S_n$으로 분할되었고 이 중 RS 부호연산으로 생성된 조각을 $S_{k+1},...,S_n$이라고 하자. (Fig.2.) RS 부호는 최소 거리 (Minimum distance) 가 $n-k+1$ 이고 $S_{k+1},...,S_n$ 의 각 심볼 값은 조각 $S_1,...,S_k$ 의 심볼들의 선형 결합 (Linear combination) 이다. $S_1,...,S_k$ 는 F 가 의사 난수치환인 CMC 알고리즘으로 암호화를 하여 생성되었으므로 그 선형 결합인 나머지 조각 $S_{k+1},...,S_n$ 도 난수와 계산적으로 구분 불가능하다. 따라서 F 에서 σ 비트의 정보를 알지 못하는 공격자에게는 모든 조각 $S_1,...,S_n$ 이 σ 에 대해 난수와 계산적으로 구분할 수 없게 된다.

임의의 조각 S_i $(1 \le i \le n)$을 가지고 있는 스토리지 서버와 Merkle 트리 증명 프로토콜을 생각해보자. 이는 Halevi 등의 방법 [6] 과 동일하다. S_i 에서 공격자가 알고 있거나 계산할 수 있는 부분의 비율을 γ $(0<\gamma<1)$ 라고 하자. Challenge로 전달되는 리프 노드의 개수를 u 라고 하면 공격자가 서버를 속일 확률, 즉 소유권 증명 프로토콜의 안정성은 Halevi 등[6]이 분석한 바와 같이 γ^u 이다. S_i 는 난수와 계산적으로 구분 불가능하므로 γ 는 매우 작은 값이다. 따라서 공격자가 공격에 성공할 확률 역시 매우 낮다.

공격자가 F 에 대한 소유권을 인정받기 위해서는 소유정보 tag_F 의 서명값 $\{tag_F\}_{sk}$ 을 가져야한다. 임계 서명의 안전성에 의해 유효한 서명값을 구하는 유일한 방법은 적어도 $t=k$개의 스토리지 서버와 프로토콜을 실행하여 서명값을 획득하는 것이다. 공격자가 t개의 스토리지 서버와 소유권 증명 프로토콜을 실행하여 모두 성공할 확률은 γ^{ut} 이다.

6.1.3 다중 CMC 암호화의 소유권 증명 안정성

4.3절에서 기술한 다중 CMC 암호화를 적용한 경우의 소유권 증명의 안정성을 분석해보자. 분석을 쉽게 하기 위해 하나의 조각 S_i $(1 \le i \le n)$ 는 ρ 개의 분할 데이터 F_j $(1 \le j \le \delta,\ \delta$는 전체 분할의 개수) 들로 이루어졌다고 가정한다. 즉, F_j의 크기 $|F_j|$는 $|S_i|=\rho|F_i|$ ($|S_i|$는 조각의 크기) 이다.

임의의 조각 S_i 에 공격자가 알지 못하는 미지의 정보를 포함한 분할 데이터 F_j 하나가 존재한다고 하자. 다중 CMC 암호화를 하게 되면 S_i 에서 공격자가 알지 못하는 부분의 비율은 $1/\rho$ 이고 나머지 $1-1/\rho$ 비율에 해당하는 부분은 알고 있다. 따라서 조각 S_i 에 대한 소유권 증명 프로토콜의 안정성은 $\gamma^u=(1-1/\rho)^u$ 이다. F_j 의 정보는 조각 $S_{k+1},...,S_n$ 으로 확산되므로 이 조각들에 대한 프로토콜의 안정성도 역시 $(1-1/\rho)^u$ 이다.

전체 n 개의 서버 중에 $t=k$ 개의 서버를 임의로 선택하여 소유권 증명을 한다고 할 때 그 중에 적어도 하나는 S_i 또는 $S_{k+1},...,S_n$ 을 가지고 있는 서버와 프로토콜을 실행해야 한다. 따라서 공격자가 소유권 증명에 성공하여 서명값 $\{tag_F\}_{sk}$ 을 구할 확률은 최대 $(1-1/\rho)^u$ 이다.

6.2 성능 분석

CMC-RS 정보분산 알고리즘의 성능을 비교하기 위하여 실험을 수행하였다. 실험을 위해 Intel i7-4770 프로세서(3.4GHz), 8GB DDR3 메모리, 그리고 1TB 하드디스크(SATA3 7200rpm)의 사양을 갖춘 PC를 사용하였으며 ext4 파일시스템의 Ubuntu 12.04 LTS 64비트 리눅스를 사용하였다. 블록암호 알고리즘과 해쉬 알고리즘으로는 각각 AES-256과 SHA-256을 선택하였고 OpenSSL 1.0.1 라이브러리를 이용하여 구현하였다. RS 부호화 알고리즘의 구현은 Jerasure 2.0 [18] 라이브러리를 이용하였다.

성능의 비교를 위하여 CAONT-RS[3] 와 CRSSS[3] 알고리즘도 동일하게 구현하였다. 모든

알고리즘에 대해 스토리지 서버의 전체 개수는 $n=16$, 가용성은 $k=8$을 설정하였으며 CRSSS 의 경우 안전성 값 r은 k의 중간값인 $r=4$로 설정하여 실험을 진행하였다. CMC-RS알고리즘은 다중 CMC 암호화를 적용한 방식도 추가로 구현하였다. 조력자의 전송 상한은 T=512MB로 하였으며 파일의 분할 크기도 T와 동일하게 하여 실험을 하였다.

Fig.5. 는 파일의 크기에 따른 알고리즘 실행 소요 시간의 측정 결과를 보여준다. CAONT-RS 와 CMC-RS 알고리즘은 $n-k$개의 조각에 대해서만 RS 부호 계산을 하는 반면 CRSSS 알고리즘은 전체 n 개의 조각들에 대해 Rabin 의 IDA계산을 해야 하므로 성능은 떨어진다. CMC-RS알고리즘은 CAONT-RS알고리즘과 비슷한 성능을 보여주고 있으나 다중 CMC 암호화를 적용한 경우 파일의 크기가 증가할수록 좋은 성능을 보여준다.

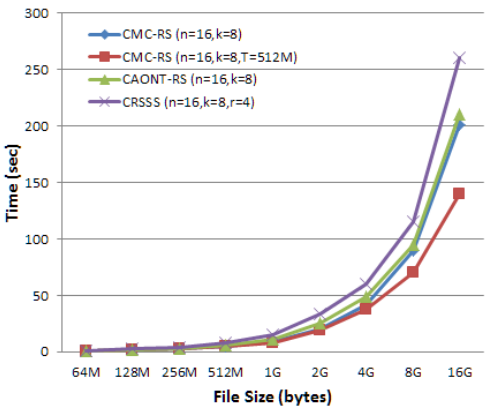

Fig. 5. Throughput of various algorithms

VII. 결 론

본 논문에서는 분산 스토리지 환경에서 클라이언트 기반 중복제거가 가능한 CMC-RS 정보분산 알고리즘과 분산 소유권 증명 기법을 제안하였다. 본 제안 방법들은 분산된 구조에서 스토리지 서버가 안전하고 신뢰할 수 있는 방법으로 데이터의 소유 사실을 검증할 수 있는 방법을 제공한다. 기존의 관련 연구에서 제안된 방법들에 비해 높은 성능을 보여주고 있으며 클라이언트 기반 중복제거가 가능하므로 스토리지 저장공간 뿐만 아니라 네트워크 대역의 효율적 절감이 가능한 장점을 제공한다.

References

[1] R. Seiger, S. Gross, and A. Schill, "SecCSIE: A Secure Cloud Storage Integrator for Enterprises," IEEE Conf. Commerce and Enterprise Computing (CEC), pp.252-255, 2011.

[2] D. Slamanig and C. Hanser, "On cloud storage and the cloud of clouds approach," International Conf. Internet Technology And Secured Transactions, pp.649-655, Dec. 2012.

[3] M. Li, C. Qin, P.C. Lee, and J. Li, "Convergent Dispersal: Toward Storage-Efficient Security in a Cloud-of-Clouds," Proc. USENIX Workshop on Hot Topics in Storage and File Systems (HotStorage'14), July 2014.

[4] Russell D., "Data deduplication will be even bigger in 2010," Gartner 2010, http://www.gartner.com/doc/1297513

[5] Harnik, Danny, B. Pinkas, and Alexandra Shulman-Peleg, "Side channels in cloud services: Deduplication in cloud storage," IEEE Security & Privacy, vol.8, no.6, pp.40-47, 2010.

[6] Halevi, Shai, D. Harnik, B. Pinkas, and A. Shulman-Peleg, "Proofs of ownership in remote storage systems," Proc. ACM Conf. Computer and communications security (CCS'11), pp.491-500, Oct. 2011.

[7] Shin Y., and Kim K., "Differentially private client-side data deduplication protocol for cloud storage services," Security Comm. Networks, doi: 10.1002/sec.1159

[8] Halevi, Shai, and Phillip Rogaway, "A tweakable enciphering mode," Advances in Cryptology-CRYPTO 2003, pp.482-499, Aug. 2003.

[9] S. Reed and G. Solomon, "Polynomial codes over certain finite fields," Journal of the Society for Industrial & Applied Mathematics, vol.8, no.2, pp.300-304, 1960.

〔10〕 A. Shamir, "How to share a secret," Communications of the ACM, vol.22, no.11, pp.612-613, 1979.
〔11〕 G. Blakley and C. Meadows, "Security of ramp schemes," Advances in Cryptology-CRYPTO 1985, pp.242-268, Aug. 1985.
〔12〕 M. Rabin, "Efficient dispersal of information for security,load balancing, and fault tolerance," Journal of the ACM, vol.36, no.2, pp.335-348, 1989.
〔13〕 J.K. Resh and J.S. Plank, "AONT-RS: Blending security and performance in dispersed storage systems," Proc. USENIX Conf. File and Storage Technologies(FAST'11), Feb. 2011.
〔14〕 R.L.Rivest, "All-or-nothing encryption and the package transform," Proc. Fast Software Encryption(FSE'97), pp.210-218, Jan. 1997.
〔15〕 M. Storer, K. Greenan, D. DE Long, and L. Miller, "Secure data deduplication," Proc. ACM International Workshop on Storage security and survivability (StorageSS'08), pp.1-10, Oct. 2008.
〔16〕 A. Boldyreva, "Threshold signatures, multisignatures and blind signatures based on the gap-Diffie-Hellman-group signature scheme," Public key cryptography(PKC), pp.31-46, 2003.
〔17〕 V. Shoup, "P ractical threshold signatures," Advances in Cryptology-EUROCRYPT 2000, pp.207-220, 2000.
〔18〕 J.S.Plank and K.Greenan, "Jerasure: A library in C facilitating erasure coding for storage applications," Tech. Report UT-EECS-14-721,Univ. Tennessee, 2014.

〈저자소개〉

사 진

신 영 주 (Youngjoo Shin) 정회원
2006년: 고려대학교 컴퓨터학과 학사 졸업(이학사)
2008년: 한국과학기술원(KAIST) 전산학과 공학석사
2014년: 한국과학기술원(KAIST) 전산학과 공학박사
2008년: LG전자 디지털미디어연구소 주임연구원
2008년~현재: 한국전자통신연구원 부설연구소 선임연구원
〈관심분야〉 암호 알고리즘, 네트워크 보안, 클라우드 보안