ISSN 1598-0170 (Print)
ISSN 2287-1136 (Online)
http://www.jksii.or.kr

하둡 분산파일시스템에서 안전한 쓰기, 읽기 모델과 평가☆

A Secure Model for Reading and Writing in Hadoop Distributed File System and its Evaluation

방 세 중* 나 일 균** 김 양 우***
Sechung Pang Ilkyeun Ra Yangwoo Kim

요 약

요즘 클라우드 컴퓨팅이 활성화됨에 따라 분산파일시스템의 요구가 증대되고 있지만 클라우드 컴퓨팅 환경에서 민감한 개인정보의 악용을 방지하는 분산파일시스템의 프레임은 아직 없다. 그래서 이 논문에서는 비밀분산 방법을 이용하여 분산파일시스템을 위한 안전한 쓰기/읽기 모델을 제시하였다. 이 모델은 비밀분산 방법을 사용하여 분산파일시스템의 기밀성뿐만 아니라 가용성도 보장한다. 또 제안한 방법으로 비밀 분산, 복구를 실행하였고 이를 대표적 암호화 알고리즘인 SEED 알고리즘에 의한 것과 비교를 함으로써 제시한 방법의 우수성을 보였다. 이와 더불어 이 방법이 하둡 분산파일시스템에 쉽게 이식될 수 있도록 하둡 분산파일시스템의 구조에 의존적이지 않은 쓰기/읽기 모델을 제안하였으며, 비밀분산모델의 성능측정방법으로 제안모델에 대한 이론적 평가를 실시하였다.

ABSTRACT

Nowadays, as Cloud computing becomes popular, a need for a DFS(distributed file system) is increased. But, in the current Cloud computing environments, there is no DFS framework that is sufficient to protect sensitive private information from attackers. Therefore, we designed and proposed a secure scheme for distributed file systems. The scheme provides confidentiality and availability for a distributed file system using a secret sharing method. In this paper, we measured the speed of encryption and decryption for our proposed method, and compared them with that of SEED algorithm which is the most popular algorithm in this field. This comparison showed the computational efficiency of our method. Moreover, the proposed secure read/write model is independent of Hadoop DFS structure so that our modified algorithm can be easily adapted for use in the HDFS. Finally, the proposed model is evaluated theoretically using performance measurement method for distributed secret sharing model.

☞ keyword : 비밀분산(Secret Sharing), 개인정보(Privacy Information), 클라우드 컴퓨팅(Cloud Computing), 하둡(Hadoop)

1. 서 론

대용량 정보처리에 대한 요구가 증대될수록 클라우드 컴퓨팅 기반의 분산파일시스템에 대한 요구는 성능과 비용뿐만 아니라 정보공유의 용이성, 지리적 제한의 최소화, 저장 공간 이용의 효율성 등, 다방면에서 지속적으로 증가하고 있다.

그리고 정보화 사회로 빠르게 진행됨에 따라 다양한 인터넷 서비스가 제공되고 있으며 이에 따라 다양한 개인정보의 요구도 증가하고 있다. 그러나 광범위하게 수집된 개인정보가 기업 또는 기관에서 관리자 등에 의한 오남용과 부주의로 노출되는 등 지속적으로 문제를 일으켜 왔다. 때때로 민감한 개인정보가 남용되거나 불법적으로 사용되어 정보 소유자에게 재산상의 손해와 정신적인 피해를 입히고 있으며 또한 기업 및 기관은 이 문제에 민사적 책임뿐만 아니라 형사적 책임까지 져야 할 경우가 있다[1].

이처럼 민감한 개인정보의 안전한 쓰기, 읽기 방법이 마련되지 않은 상태에서 분산파일시스템을 사용하는 것은 개인정보 관리라는 정보보호 측면에서 많은 문제를 일으킬 수 있다. 따라서 이 논문에서는 클라우드 컴퓨팅 환경의 분산파일시스템(Distributed File System: DFS)으로 주목을 받고 있는 Hadoop DFS에 기밀성을 보장하여 개인정보를 안전하게 저장할 수 있는 분산모델을 소개한

* 정 회 원 : KCC시큐리티 정보보호연구소 연구소장
road2@kccsecurity.com
** 정 회 원 : 미국 콜로라도 주립대학교 (덴버 캠퍼스) 컴퓨터공학과 부교수
Ilkyeun.Ra@ucdenver.edu
*** 종신회원 : 동국대학교 정보통신공학과 교수(교신저자)
ywkim@dongguk.edu
[2012/07/10 투고 - 2012/08/03 심사 - 2012/09/27 심사완료]
☆ "본 연구는 지식경제부 및 정본통신산업진흥원의 대학 IT연구센터 지원사업의 연구결과로 수행되었음" (NIPA-2012-H0301-12-3006)
☆ "이 논문은 2012년도 정부(교육과학기술부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(No. S2012A010400001)."

다. 이 방법은 Shamir의 비밀분산(Secret Sharing) 기법을 수정한 것으로 침해당한 데이터 서버에 있는 정보로부터 중요하고 민감한 개인정보를 추론하는 것을 방지하며, 동시에 분산파일시스템의 중요 특징인 가용성도 같이 보장한다. 또한 정보의 기밀성을 보장하기 위한 방법 중 하나인 대칭키 암호화 알고리즘인 SEED[2]의 암호화, 복호화 성능과 여기서 제안된 알고리즘의 성능을 같은 조건에서 비교하였다. 또한 HDFS 구조를 변경하지 않고도 제안한 방법이 독립적으로 작동할 수 있는 안전한 쓰기/읽기 구조를 제안하였고 이 구조를 비밀분산의 성능측정 방법을 사용하여 이론적으로 분석하였다.

2. 관련연구

기존의 관련연구는 크게 둘로 나눌 수 있는데 이들 중, 하나는 개인정보를 담고 있는 데이터를 활용할 목적으로 배포할 때 민감한 개인정보가 유출되지 않도록 하는데 초점을 맞춰 개발되었다. 따라서 개인정보에 민감한 정보를 삭제한 후 배포하는 탈-식별화(de-identification) 방법이 제시되었다. 그러나 이 방법은 배포된 정보와 이 정보와 연결된 데이터를 통해 개인정보를 추론할 수 있는 추론공격이 가능하다. 그래서 프라이버시 보호 방법으로 익명화(anonymization) 방법인 k-anonymity[3], l-diversity[4]를 비롯한 왜곡(distortion), 압축(condensation), 교환(swapping), 분해(anatomy) 등의 방법이 연구되었다. 그러나 이러한 기법들은 주로 데이터베이스 환경을 대상으로 연구가 이루어지고 있고 텍스트 기반의 정보에 대한 접근은 상대적으로 이루어지지 않았다.

그리고 또 하나는 정당한 권한이 없는 사용자에게 개인정보가 노출되지 않도록 하기 위한 암호화 방법으로 이 알고리즘은 두 방법으로 나뉜다. 즉, 대칭키 암호화 방법과 비대칭키 암호화 방법이다. 대표적인 대칭키 암호화 알고리즘은 DES[5]인데 이 알고리즘은 1976년에 미국 표준이 되었고 56비트 키 크기를 갖고 있다. 또 128, 192, 256비트 키 크기를 갖는 AES[6] 알고리즘은 2001년에 새로운 미국 표준이 되었다. 그리고 한국에서는, 128비트를 갖는 SEED[7] 알고리즘이 1999년 표준으로 제정이 되었고 2005년 개정이 되었다. 또한 하나의 쌍인 서로 다른 두 개의 키를 갖는 비대칭 키 알고리즘의 전형적인 알고리즘은 RSA[8] 알고리즘이다. 이 암호화 방법은 하나의 키로 평문을 암호화하여 만든 암호문을 다시 평문으로 만들기 위해서 암호화 할 때 사용한 키의 쌍인 다른 키만 사용할 수 있다. 즉, 대칭키 암호화 방법은 암호화하거나 복호화할 때 동일한 키를 사용하지만 비대칭 키 암호화 방법은 서로 다른 키를 이용하여 암호화하거나 복호화 한다. 정보보호 분야에서, 암호화 알고리즘 개발 분야는 가장 광범위하게 연구되어 왔고 그만큼 적용분야도 많다. 그러나 암호화 알고리즘은 송신자와 수신자 양측에서 암/복호화가 진행되기에 정보가 여러 곳에 나누어져 있는 분산 구조에서 가용성을 보장하는 분산파일시스템에서는 적합하지 않다.

그런데 기존 방법들과 다르게 비밀분산법(secret sharing scheme)[9-10]이 적용된 모델은 그 구조적 특징으로 인해 기밀성뿐만 아니라 가용성을 모두 보장한다. 이 논문에서는 암호 알고리즘의 비밀키를 안전하게 관리하는 방법인 Shamir의 비밀분산 방법 즉, (t,n) 임계 값 방법(threshold method)[9]을 사용한다. 여기서 t,n은 $t \le n$의 관계를 갖고 있고 모두 양의 정수이며, 전체 참여자 집합을 P라고 할 때 P중에서 n명의 참여자들인 P_i에게 임의의 키인 비밀정보 K를 다항식을 통해 변형된 값(인) S_{P_i}(비밀조각: secret share)들로 분배한다면, 임의의 t명 이상의 참여자는 자신들이 갖고 있는 값으로 비밀정보 K를 구할 수 있지만, $t-1$명 이하의 참여자가 모여서는 정확한 K를 구할 수 없는 구조이다. 이 방법에서 사용하는 다항식은 식 (1)과 같다.

$$f(x_i) = K + \sum_{j=1}^{t-1} a_j x_i^j \bmod q \qquad (1)$$

여기서 $q \ge n+1$이며 소수이다.

이와 더불어 (t,n)임계 값 방법을 일반화하여 참가자 집합 P에 대해 참가자 t명으로 이루어진 부분집합을 접근구조(access structure)라 하고 이것을 구성하는 방법에 따라 다양한 비밀분산방법이 존재할 수 있다. 이때, 정확히 임계 값 t명 이상으로 이루어진 비밀정보 참가자들의 부분집합은 자신들이 갖고 있는 비밀조각을 이용하여 비밀정보 K를 복원할 수 있지만, t명 미만의 참가자들의 부분집합은 결코 어떤 정보도 획득할 수 없다고 하여 이를 완전 비밀분산법(perfect secret sharing scheme)이라고 한다[11].

따라서 임의의 접근구조에 대한 다양한 비밀분산 방법이 존재할 수 있으며 최적화된 방법을 구분하기 위한 접근구조의 성능 측정방법이 제시되었다. 즉, 제시된 시스템의 안정성은 참여자들에게 분산되는 정보량에 반비례한다는 것이다. 그러므로 감추고자 하는 비밀정보량에 대한 분배되는 비밀조각의 정보량의 비율인 정보비

(information rate)[12]를 비밀분산법의 주요 성능 측정 기준으로 삼는다. 정보비는 아래와 같이 정의된다.

$$\rho = \frac{\log_2 q}{\log_2 s} \quad (2)$$

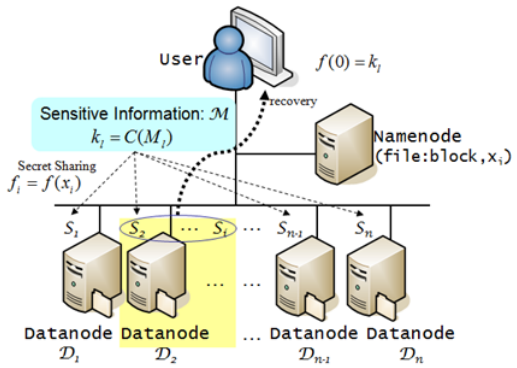
여기서 q는 비밀정보 K가 가질 수 있는 모든 가능한 값의 개수이고, s는 정보를 분산할 수 있는 가능한 집합의 최대 크기이다. 즉, 식 (2)의 의미는 비밀정보 K가 가지는 비트의 길이와 분할된 분산정보가 갖는 비트 길이의 비율이다. 즉, 식 (2)의 값은 $\rho \leq 1$이고 $\rho = 1$인 비밀분산 방법을 이상적 비밀분산법이라고 한다[13]. 따라서 Shamir의 비밀분산법은 이상적 비밀분산법이라는 것을 알 수 있다.

그리고 이런 비밀분산법은 여러 목적에 따라 다양한 형태로 변형되어 제시되었다. 참여자들의 비밀조각에 대한 접근권한이 동일하지 않은 모델을 지원하는 구조[14-15], 계산해야 되는 비밀조각이 많은 경우 고속으로 비밀분산과 비밀복구를 지원하는 구조[16], 그리고 비밀조각으로 만들어야 되는 데이터의 양이 큰 경우 이를 관리하는 구조[17] 등 여러 형태로 적용되어 왔다.

3. 제시모델

클라우드 컴퓨팅 기반의 분산파일 시스템(DFS)은 효율적인 컴퓨팅 자원의 활용과 그에 따른 비용절감뿐만 아니라 대규모로 저장되어 있는 수집된 정보를 쉽게 이용할 목적을 갖는다. 그러나 앞서 언급한 바와 같이 평문 형태로 데이터 서버에 개인정보가 분산되어 저장된다면, 정보가 부분적으로만 누출된다고 하더라도 추론공격으로 인하여 신용카드 또는 병력 정보와 같은 민감한 개인정보의 유출로 이어질 수 있다. 따라서 본 논문에서는 DFS를 사용하는 분산 시스템에 민감한 정보를 변형하여 나누어 저장함으로써 정보가 일부 누출되어도 추론공격을 방어할 수 있는 모델을 제안하였다. 또한 일부 데이터 서버를 사용할 수 없거나 접속할 수 없는 경우에도 임계값 이상으로 변형하여 나누어진 정보들을 모을 수 있다면 수집된 정보들로부터 원래의 민감한 정보를 복원할 수 있다.

이와 더불어 기존 Shamir의 임계 값 방법에서 사용된 감출 키의 길이는 56비트, 128비트, 192비트, 256비트이지만[2],[5-7],[18], 본 논문에서 제안하는 방법은 임의의



(그림 1) 수정된 Shamir의 비밀분산 방법이 적용된 DFS의 개념도

암호키 대신에 보호대상인 민감한 개인정보를 적당한 크기로 나누어 다항식의 상수항에 대응하도록 확장하였다.

또한 기존방법처럼 비밀조각 $f(x_i)$와 x_i를 같이 보관하는 것은 일부 데이터 서버에서 노출된 정보가 무차별 대입 공격(Brute Force Attack)을 통해 비밀조각을 계산한 완전한 다항식을 발견할 수 있기 때문에 본 논문에서는 기밀성을 구조적으로 보장할 수 있도록 분산파일시스템의 구조를 효율적으로 사용하였다. 즉, x_i 값은 분산파일시스템의 파일 이미지 테이블에 저장하고, 비밀조각 $f(x_i)$는 분산파일시스템의 데이터 서버에 분할하여 저장되도록 하여 쉽게 무차별 대입 공격 시간을 줄일 수 없도록 하였다. 그리고 하둡 DFS(HDFS)[19]는 대용량의 파일 관리를 지원하고 다중 파일 복사본을 저장하여 가용성을 보장하는 구조를 갖고 있는데, 파일 복사본을 갖고 있는 데이터노드(Datanode)들뿐만 아니라 파일 이미지 정보를 갖는 네임노드(Namenode)를 갖는 구조이다[20-21]. 그래서 3.3 절에서는 네임노드와 데이터노드에서 비밀을 분산하고 복구하는 과정을 HDFS의 파일 읽기와 쓰기 과정에 맞게 추가한 방법을 제시하였다. 그림 1은 위에서 설명한 개인정보 비밀 분산/복구 과정을 묘사한다.

3.1 개인정보의 비밀분산 저장 방법

구체적으로 수정된 (t,n)비밀분산 저장 방법은 다음과 같이 기술할 수 있다.

- 입력: ① 비밀조각을 계산할 다항식의 차수$(t-1)$, ② 비밀스트립 S_i가 저장될 데이터노드의 총 개수

(n), ③민감한 개인정보 텍스트(M)

• 출력: ① 비밀값(S_{il}), ② 비밀값을 계산한 x_i값

1) 텍스트 형식의 개인정보 M을 이진수 $k_l = C(M_l)$으로 변환한다.

$M_l \in M,\ M = M_1, M_2, ..., M_L,\ 1 \le l \le L$

여기서 L은 조각난 개인정보 M_l의 전체 나누어진 수이다.

- 조건: $M = M_1 \cup M_2 \cup ... \cup M_L$

$\Phi = M_1 \cap M_2 \cap ... \cap M_L$

k_l가 양의 정수가 되도록 M을 적당한 L로 나누어 M_l을 만든다.

2) 임의의 소수 $q(q \ge n+1)$를 선택한다.

여기서 n은 총 데이터노드의 수이다

3) 서로 다른 각각의 임의의 계수 값들과 임의의 x_i 값을 선택한다.

$$a_1, a_2, ..., a_{t-1} (a_j \in Z_q, 1 \le j \le t-1),$$
$$x_i (1 \le i \le n) \quad (3)$$

여기서 $t-1$은 식(4)의 다항식의 차수이다

4) 식(4)와 같이 변형된 개인정보 k_l을 상수항으로 하는 임의의 $t-1$ 차수의 다항식을 선택한다.

$$f_l(x_i) = k_l + \sum_{j=1}^{t-1} a_j x^j \bmod q \equiv S_{il} \quad (4)$$

5) x_i값에 대한 비밀조각(비밀값) $S_{il} = f_l(x_i)$를 계산한다.

6) 모든 $l(1 \le l \le L)$에 대해 3~5 단계를 반복하여 식(5)의 비밀스트립(secret strip)을 얻는다.

$$S_i = \{S_{i1}, S_{i2}, ..., S_{il}, ..., S_{iL}\},\ (x_i, S_i) \quad (5)$$

7) 모든 $i(1 \le i \le n)$의 S_i에 대해 이전의 5번과 6번 단계를 반복하여 비밀조각들의 집합 S_l을 구한다.

$$S_l = \{S_{1l}, S_{2l}, ..., S_{il}, ..., S_{nl}\} \quad (6)$$

8) 계산된 모든 S_{il}에 대해 일정한 구조를 갖는 비밀파일(secret file) S을 만든다(3.3절에 상세하게 기술).

9) 비밀스트립 쌍 (x_i, S_i)의 x_i값을 (그림 4)의 c와 같이 블록 복제 구성 파일에 확장하여 기록한다.

3.2 개인정보의 비밀복구 방법

앞서 설명한 방법으로 변형되어 분산 저장된 개인정보의 비밀복구 방법은 다음과 같다.

• 입력: ① t 개 이상의 비밀값(비밀조각), ② 이에 대응하는 x_i값

• 출력: ① 민감한 개인정보 텍스트(M)

1) 데이터노드에 변형되어 분산 저장된 개인정보 파일과 네임노드에서 x_i을 읽어 들인다.

2) 비밀파일로부터 비밀스트립 S_i을 읽어 들인다.

3) 읽어 들인 비밀스트립 S_i에 *대응하는* 모든 x_i값들을 찾는다.

4) 재조합한 파일로부*터 t 이상의 S_i*를 추출한다(식6).

5) 비밀조각 $S_{il} = f_l(x_i)$과 대응 값 x_i들을 사용하여 라그랑지 보간법(Lagrange's Interpolation)을 통해 분할된 개인정보의 이진수 $f_l(0) = k_l$를 구한다.

$$f_l(0) = \sum_{i=1}^{t} S_{il} \prod_{j=1, j \ne i}^{t} \frac{x_j}{x_j - x_i} \bmod q \quad (7)$$

6) 이진 개인정보 k_l을 분할된 개인정보 $M_l = C^{-1}(k_l)$로 변환한다.

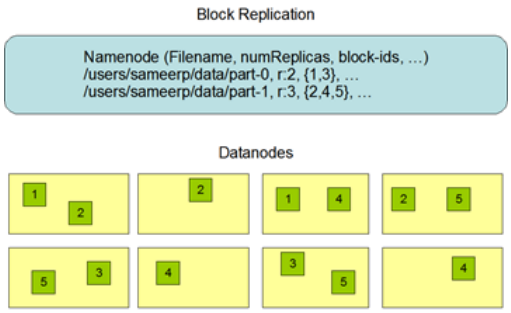
7) 모든 $l(1 \le l \le L)$에 대해 5~6번 단계를 반복한다.

8) 모든 분할된 개인정보 M_l을 개인정보 $M = M_1 \cup M_2 \cup ... \cup M_L$로 합한다.

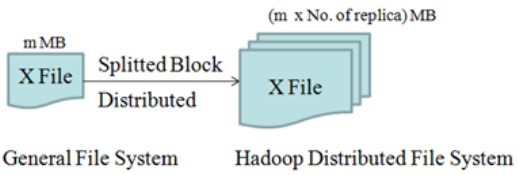
위의 라그랑지 보간법 대신에 일반적인 행렬 방정식을 사용할 수도 있으나 일반적으로 정보를 복원하는데 시간이 더 소요된다[22].

3.3 HDFS의 수정 모델

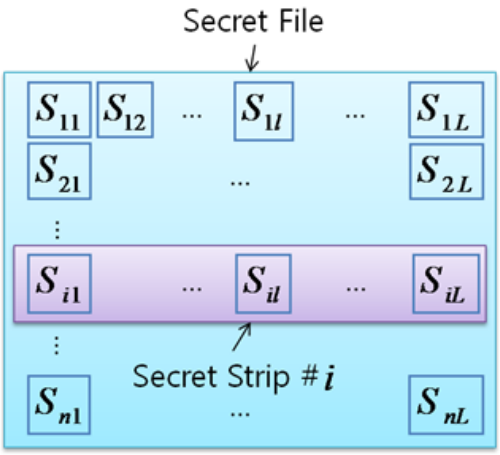
앞서 설명한 비밀분산 알고리즘을 통해서 얻은 수정된 정보를 HDFS의 구조 변경 없이 쓰기/읽기 위해서는 추가적인 계층이 필요하다. 3.1 절에서 기술한 방법에 따라 일반 파일시스템에서 만들어진 비밀파일을 HDFS로 전송하면 (그림 2)의 b와 같이 해당 비밀파일은 복사본의 수자와 기본 파일블록 크기로 나뉘고 (그림 2)의 a에서 볼 수 있는 것처럼 HDFS의 기본 파일 블록 크기보다 큰 파일은 모두 같은 크기의 블록으로 나뉘어 HDFS의 많은 데이터노드에 분산 저장된다.



(그림 2-a) HDFS 파일블록의 분산 저장 개요 [20]

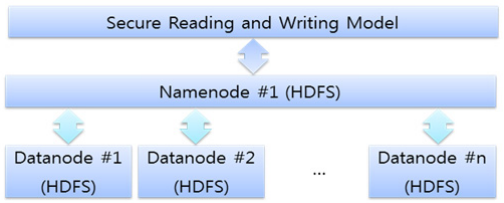


(그림 2-b) HDFS 파일블록의 분산 저장 프로세스 개념

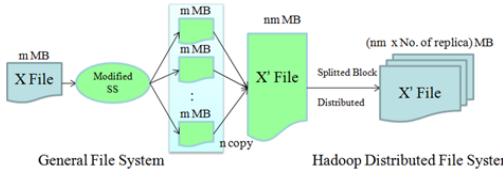


(그림 3) 비밀조각(secret share), 비밀스트립(secret strip) 및 비밀파일(secret file) 간의 관계

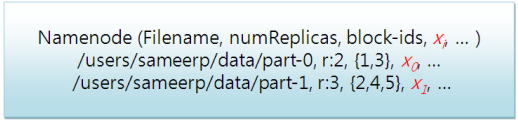
따라서 (그림 3)과 같이 만들어진 비밀조각 S_{il}를 모아 $S_i = \{S_{i1}, S_{i2}, ..., S_{il}, ..., S_{iL}\}$를 만든다. 이를 비밀스트립(secret strip)이라 정의한다. 이때 만들어진 비밀스트립 S_i의 크기는 HDFS의 파일 블록의 크기와 동일하게 한다. 그리고 3.1 절에서 기술한 것처럼 분배할 데이터노드의 수만큼 이 비밀스트립을 만들고 이 비밀스트립을 다시



(그림 4-a) 기존 HDFS에 독립적인 안전한 쓰기/읽기 모델 개념도



(그림 4-b) 제안한 안전한 쓰기/읽기 모델의 HDFS 적용 개념



(그림 4-c) 네임노드 블록 복제 구성파일 확장

하나의 파일로 만들고 비밀파일(secret file)이라고 부른다.

또한 (그림 4)의 c처럼 비밀스트립 위치정보와 함께 비밀 분산/복원에 필요한 x_i정보를 같이 저장한다. 그러면 이 비밀파일은 (그림 4)의 b와 같이 HDFS의 위 계층인 안전한 쓰기/읽기 계층에서 HDFS로 전송되면서 비밀스트립 단위 즉, HDFS의 파일블록 크기로 나뉘어 데이터노드에 분산 저장되는 것이다.

이와 유사하게 HDFS에서 파일을 읽어 오는 과정은 역순으로 이루어진다. 데이터노드로부터 분산 저장되어 있는 파일블록을 읽어 파일로 조합해 안전한 쓰기/읽기 계층으로 전송하면 하나의 비밀스트립으로 재조합 되고, 복호화 가능한 임계 값 이상의 비밀스트립들을 읽어 이것을 3.2 절에 기술된 절차에 따라 민감한 개인정보로 복원한다. 지금까지 기술한 비밀분산법을 이용한 안전한 쓰기/읽기 하둡 분산파일시스템(HS3: HDFS using Secret Sharing Scheme)의 전체 구성도는 (그림 4)의 a와 같이 표현할 수 있다.

4. 시험 및 평가

4.1 SEED 알고리즘과 비밀분산 알고리즘의 성능 비교

4.1.1 실험환경

HDFS의 기본 파일 블록의 크기는 64MB이다[21]. 따라서 알고리즘 비교에 사용되는 데이터 크기는 64MB로 정했다. 그리고 비교 대상인 SEED[2],[7],[18]의 구현 모듈은 한국인터넷진흥원(KISA)에서 제공하는 Java 코드의 표준모듈을 사용하였다. 또한 제안한 비밀분산 알고리즘 구현의 개발도구로는 Eclipse V3.5.2[23]와 Java 버전 jre 1.6.0_19[24]를 사용하여 개발하였다. 테스트 환경은 CPU는 Intel Core2 Quad 2.83GHz, 메모리는 2GB, OS는 Windows7(32bit)이다. 이때 사용한 테스트 벡터의 크기는 512비트, 1024비트로 블록암호알고리즘 SEED의 운영모드[18]에서 제시한 값을 사용하였다. SEED 알고리즘과 수정된 비밀분산 알고리즘은 모두 위와 같은 동일 시스템에서 수행되었고, 이번 실험에서 비밀분산 알고리즘에 사용된 다항식은 2차 다항식을 기준으로 하였다. 또 조각난 개인정보 M_i의 크기는 32비트(4바이트)가 되도록 하였다. 이것은 구현 시 Java 정수형의 크기가 4바이트이기에 편의적으로 정하여 비밀분산, 복구 알고리즘을 구현한 것이다. 그러나 Java의 다른 클래스[25]를 이용하면 이보다 큰 크기의 개인정보를 만들 수도 있다.

4.1.2 실험결과

10회 반복하여 측정한 평균값을 보여주는 (표 1)과 (표 2)에서 알 수 있듯이 64MB 파일을 암호화하거나 비밀조각을 구하는데 테스트 벡터의 크기에 대한 영향은 거의 없었다. 즉, 두 방법 모두 대상 텍스트의 크기는 암/복호화 또는 비밀분산 방법의 수행속도에 영향을 거의 주지 않는다고 말할 수 있다. 여기서 구한 비밀조각 3개는 2차 다항식을 사용하였기에 비밀 복원하는데 필요한 최소 개수이다.

그러나 제시한 모델에서 비밀분산을 위해 2차 다항식을 계산하는 속도는 SEED의 경우와 대비하여 54%정도 빠르지만, 라그랑지 보간법으로 비밀 복원하는데 계산하는 속도는 SEED 보다 대략 20% 정도만 빠르다는 것을 알 수 있었다. 또한, SEED 알고리즘으로 암/복호화 하는데 걸리는 수행시간 차이는 거의 나지 않지만, 본 논문에서 제안한 방법(HS3)으로 비밀분산/복원하는 경우의 수행시간은 79%정도의 차이로 비밀분산이 비밀복원보다 빠르다는 것을 알 수 있다. 즉, 제안한 방법이 전체적으로 기존 SEED 알고리즘보다 수행시간이 짧다는 것을 확인할 수 있었다. 추가적으로 512비트 테스트벡터를 사용하여 20회 반복하여 측정한 평균값과 표준편차는 (표 3)에서 볼 수 있다. (표 2)의 값과 비교하면 평균값이 소수점 첫째자리까지 변화가 없었다.

(표 1) SEED와 비밀분산 방법의 성능비교 (512비트 테스트 벡터)

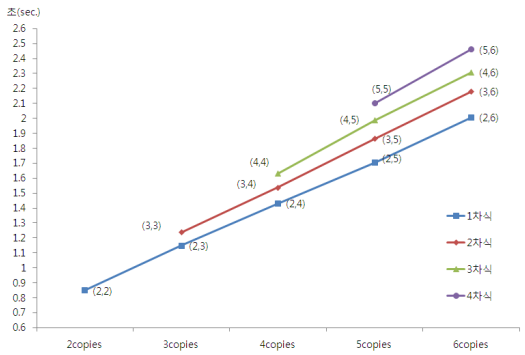
Method		Average(sec.)
SEED (1copy)	Encryption	2.6965
	Decryption	2.753
Threshold Secret Sharing (3 secret shares)	Secret Sharing	1.2382
	Secret Recovery	2.216

(표 2) SEED와 비밀분산 방법의 성능비교 (1024비트 테스트 벡터)

Method		Average(sec.)
SEED (1copy)	Encryption	2.7648
	Decryption	2.757
Threshold Secret Sharing (3 secret shares)	Secret Sharing	1.23
	Secret Recovery	2.189

(표 3) SEED와 비밀분산 방법의 성능비교 (512비트 테스트 벡터)

Method		Average (sec.)	Standard dev.
SEED (1copy)	Encryption	2.684	0.0327
	Decryption	2.734	0.0385
Threshold Secret Sharing (3 secret shares)	Secret Sharing	1.234	0.1677
	Secret Recovery	2.202	0.0431



(그림 5) (t,n) 비밀분산 성능: 다항식의 차수(t)와 비밀분산 조각(n) 개수에 따른 시간

4.2 비밀분산 알고리즘의 비밀 분산 처리 속도 비교

4.2.1 실험환경

실험환경은 4.1.1 절에서 기술한 내용과 동일하며 실험 절차는 1차식에서부터 4차 다항식까지에 대해 2개에서 6개의 비밀조각을 만들면서 테스트를 진행하였다. 이때 사용한 *테스트 벡터*의 크기는 512비트만 사용하였다.

4.2.2 실험결과

(그림 5)는 (t,n) 임계 비밀분산 방법에서 차수 n이 1차식부터 2차, 3차, 4차 다항식으로 차수가 증가하고, 각각의 임계 값 t의 값은 각각 2~5부터 시작하여 최종 6개의 비밀조각을 만드는데 소요되는 시간을 측정한 것이다. 차수가 증가할수록, 또 비밀조각의 개수가 증가할수록 처리시간은 증가하지만 각 차수의 다항식에 대해 처리속도의 기울기를 결정하는 것은 해당 다항식의 차수만 영향을 미친다는 것을 알 수 있었다. 그리고 우리가 측정한 범위 내에서 비밀조각 생성시간은 SEED 알고리즘의 의한 암호 생성시간보다 작음을 확인하였다.

4.3 제시한 비밀분산법의 성능분석

4.3.1 비밀파일의 정보비

앞서 식(2)에서 보았듯이 이진 개인정보 k_l가 가질 수 있는 값의 크기는 4바이트(q의 비트 길이)이고 이것을 갖고 만드는 비밀조각의 크기도 4바이트이지만 3.3 절에 기술한 바와 같이 비밀조각을 모아 데이터노드에 분산 저장하는 비밀스트립 S_i의 크기는 HDFS의 기본 파일블록 크기로 64MB이다. 따라서 이들의 각각의 비트 수는 5와 29이다. 이를 정리하여 정보비(information rate)를 계산한 것이 (표 4)이다.

(표 4) 실험조건(Java Int형 크기와 HDFS의 기본 파일블록 크기)에 대한 정보비

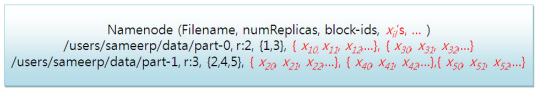
q	$\log_2 q$	s	$\log_2 s$	$\rho = \frac{\log_2 q}{\log_2 s}$
2^5(=4Bytes)	5	*64MB	29	0.172414

4.3.2 접근구조 분석

비밀분산법이 적용된 모델은 그 구조적 특징으로 인해 기밀성뿐만 아니라 가용성을 모두 보장하기에 해당 구조를 이용한 본 논문의 HS3는 기밀성 및 가용성을 만족한다. 그리고 감추고자 하는 비밀정보량에 대한 분배되는 비밀조각의 정보량의 비율인 정보비를 비밀분산법의 주요성능 측정 기준으로 삼는데 완전 비밀분산법인 Shamir의 비밀분산법을 적용하여 제안된 모델의 정보비를 계산하면 약 0.17정도의 작은 값이 나온다. 그 이유는 HDFS의 구조를 변형시키지 않고 모델을 적용한 이유와 프로그램 편의성 때문이다. 즉, 비밀스트립 내의 비밀조각을 계산하는데 동일한 x_i값을 사용하였기 때문에 기본 파일블록의 크기가 64MB인 비밀스트립은 하나의 x_i값으로 해당 비밀스트립에 속한 모든 비밀조각들을 계산하고, 이와 더불어 이진 개인정보의 최대크기를 4바이트(Java 정수형 크기)로 했기 때문이다.

따라서 식 (2)의 정보비에서 s값을 작게 하던지 q값을 크게 하면 임계 값 t 미만에서 얻는 비밀스트립으로부터는 어떤 정보도 복원할 수 없는 완전 비밀분산법에 가까워질 것이다. 구체적으로 s값을 작게 하는 방법은 비밀스트립 즉, HDFS 파일블록의 크기를 작게 하면서 그에 따른 비밀조각을 만드는 x_i값을 모두 다르게 하는 방법이다. 이 방법의 단점은 파일블록의 크기가 1/2로 작아질수록 분산 저장해야 되는 데이터노드의 수가 2배 많아져야 하며 (그림 6)에서 볼 수 있는 것처럼 블록정보를 담고 있는 파일이 서로 다른 값들을 모두 저장하기 위해 커진다는 것이다. 즉, 파일블록과 이진 개인정보의 크기가 각각 64MB와 4바이트이고 또 비밀조각 x_i값의 크기가 4

* 64MB=2^6 x 2^{20} x 2^3 = 2^{29}



(그림 6) 정보비 증가를 위한 네임노드 블록 복제 구성파일의 확장 예시

바이트라고 하면 블록 복제 정보를 담고 있는 파일의 크기는 약 64MB만큼 커질 수 있다는 것이다.

또 q값을 크게 해서도 정보비를 증가시킬 수 있는데 이는 현재 제시된 틀에서 Java의 큰 정수를 다루는 클래스[25]를 이용하여 구현할 수 있다. 이론적으로는 무한히 큰 정수를 다룰 수 있지만 컴퓨터 시스템 자원의 한계로 현재 s값인 64MB만큼 큰 수(2^{64MB})를 다룬다는 것은 어려운 일이다. 그러므로 두 경우를 고려하여 참여시킬 수 있는 데이터노드의 수와 비밀조각을 계산하는 시스템의 컴퓨팅 자원을 고려하면 정보비 ρ를 많이 향상시킬 수 있다. 한 예로, 파일블록(비밀스트립) 크기를 2MB 이하, 비밀조각의 크기를 512바이트 이상으로 했을 때 정보비 ρ는 0.5 이상의 값을 얻을 수 있다.

5. 결 론

본 논문에서 제안한 분산파일시스템용 안전한 쓰기/읽기 모델(HS3)은 다음과 같이 4가지 특징을 갖고 있다. 먼저, 정보의 부분 유출로 인한 추론공격의 가능성을 차단하고, 둘째, 대표적인 SEED 알고리즘과 비교해서 평균적으로 높은 성능을 보이며, 셋째, HDFS의 데이터노드 중 일부를 접근할 수 없다고 할지라도 (t,n) 임계 값 방법과 관련하여 t 이상의 데이터노드를 사용할 수 있다면 개인정보를 복원할 수 있다. 마지막으로, 제안한 안전한 HDFS 파일 쓰기/읽기 모델은 기존 HDFS의 구조 변경 없이 독립적으로 수정된 비밀 분산/복원 방법을 적용할 수 있게 한다는 것이다.

지금까지 개인정보의 안전한 분산저장이 가능하며 대표적인 대칭키 알고리즘에 비교해도 효율적인 분산파일시스템용 쓰기/읽기 모델을 제안하였으며, 또한 이 모델은 HDFS에 독립적이기 때문에 기존 HDFS에 쉽게 적용할 수 있다. 그러나 평문인 개인정보 파일의 크기는 암호문만 만들 경우에는 변하지 않지만 이 논문에서 제시된 방법을 사용하면 임계 값 t 이상의 크기로 전체 분량은 커진다. 하지만 HDFS가 기본적으로 3개의 복사본을 유지하는 것처럼 일반적인 분산파일시스템들도 가용성을 보장하기 위해 복사본을 여러 개 두는 것을 고려하면 제시된 방법이 기밀성과 가용성을 동시에 만족시킨다는 관점에선 저장 공간을 추가로 더 사용하지 않는다고 할 수 있다. 또한 정보비 ρ가 1 미만의 값을 갖는다는 것은 t 미만의 비밀스트립이 노출되면 무차별 대입 공격을 통한 추론의 가능성이 발생한다는 것이다. 이 가능성을 제거하기 위해서는 비밀스트립 쌍인 (x_i, S_i)의 x_i를 비밀스트립을 구성하고 있는 비밀조각의 개수 L만큼 서로 다른 값을 이용하여 비밀정보를 적당한 크기의 비밀조각으로 만들고 비밀조각으로 구성된 비밀스트립 개수만큼의 데이터노드에 나누어 저장한다면 정보비 ρ가 많이 향상된 비밀분산법이 될 수 있다. 그러나 이때, 메모리에 탑재되는 HDFS 네임노드의 메타파일(그림 6의 내용)의 크기가 추가적으로 커질 것이다.

이제까지 완전 비밀분산법인 Shamir의 비밀분산 방법을 클라우드 컴퓨팅용 HDFS에 적용하여 민감한 개인정보의 기밀성 및 가용성 제공과 HDFS에 독립적인 안전한 읽기, 쓰기 모델을 제안하였다. 향후 연구방향은 데이터노드로부터 노출된 개인정보의 일부를 가지고 추론 및 무차별 대입 공격을 하더라도 비밀분산에 사용된 다항식을 추정하기 어렵도록 접근구조를 견고하게($\rho \to 1$) 하며, 동시에 HDFS의 안정성도 떨어뜨리지 않는 향상된 모델로 개선할 예정이다.

참 고 문 헌

[1] 한국인터넷진흥원 개인정보보호, http://privacy.kisa.or.kr

[2] SEED 공식 사이트, http://seed.kisa.or.kr/seed/jsp/seed.jsp

[3] L. Sweeney, "k-anonymity: a model for protection privacy", *International Journal on Uncertainty, Fuzziness and Knowledge-based Systems*, 10(5), 2002. pp.557-570.

[4] A. Machanavajjhala, J. Gehrke, and D. Kifer, "l-Diversity: Privacy beyond k-anonymity". *In International Conference on Data Engineering (ICDE)*, 2006. pp.24-35.

[5] DES(Data Encryption Standard), http://csrc.nist.gov/publications/fips/fips46-3/fips46-3.pdf

[6] AES(Advanced Encryption Standard), http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf

[7] Telecommunications Technology Association, 128-bit Block Cipher SEED (TTA.KO-12.0004/R1), 1999.

[8] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems," *Communications of ACM*, Vol. 21, pp. 120-126, 1978.

[9] A. Shamir, "How to share a secret", *Communications of the ACM*, vol.22, 1979. pp.612-613.

[10] Blakley, G. R. "Safeguarding cryptographic keys", *Proceedings of the National Computer Conference 48*, pp.313 - 317, 1979.

[11] J. Benaloh, J. Leichter, "Generalized Secret Sharing and Monotone Functions", *In Advances in Cryptology-CRYPTO '88, Lecture Notes in Computer Science*, vol. 403, pp. 27-35, 1990.

[12] C. Blundo, A, De Santis, L. Gargano, and U. Vaccaro, "On the Information Rate of Secret Sharing Schemes", *Theoretical Computer Science*, vol. 154(2), pp. 283-306, 1996.

[13] E. F. Brickell and D. M. Davenport, "On the Classification of Ideal Secret Sharing Scheme", *Journal of Cryptology*, vol. 4, pp. 123-134, 1991.

[14] 박소영, 이상호, 권대성, "가중치를 갖는 비밀분산법", *정보과학회논문지: 시스템 및 이론* 제29권 제4호, 2002.4.

[15] Oriol Farras, Jaume Martí-Farré , Carles Padró, "Ideal Multipartite Secret Sharing Schemes", Advances in Cryptology, Eurocrypt 2007, *Lecture Notes in Computer Science*, Vol.4515, pp. 448-465, 2007.

[16] Jun Kurihara, Shinsaku Kiyomoto, Kazuhide Fukushima and Toshiaki Tanaka, "A New (k, n)-Threshold Secret Sharing Scheme and Its Extension", *Lecture Notes in Computer Science*, Vol. 5222, pp.455-470, 2008.

[17] 송유진, 이동혁, "New Secret Sharing Scheme for Privacy Data Management", *한국정보보호학회 2006년도 하계학술대회*, Vol.16, No.1, pp.765-773, 2006.

[18] Telecommunications Technology Association, Modes of Operation for the Block Cipher SEED (TTAS. KO-12.0025), 2003.

[19] Hadoop Official Site, http://hadoop.apache.org

[20] HDFS Architecture, Hadoop 0.20 Documentation, http://hadoop.apache.org/common/docs/r0.20.2/hdfs_design.html

[21] Tom White, *Hadoop, The Definitive Guide*, 3. The Hadoop Distributed Filesystem, Data Flow, Ch.3, O'REILLY, 2009.

[22] 방세중, 조성환, 이승하, 김양우, "클라우드 컴퓨팅 환경에서 안전한 개인정보 분산저장 모델 및 평가", *한국인터넷정보학회 10주년 기념학회*, Vol.11 No.1, pp.169-170, 2010.

[23] Eclipse 공식 사이트, http://www.eclipse.org

[24] Java 공식 사이트, http://www.java.com

[25] Java.math Class BigInteger 공식 사이트, http://download.oracle.com/javase/1.4.2/docs/api/java/math/BigInteger.html

◐ 저 자 소 개 ◑

방 세 중

1993년 서강대학교 물리학과 졸업(학사)
1995년 서강대학교 대학원 물리학과 졸업(석사)
2003년 동국대학교 산업대학원 정보통신공학과 졸업(석사)
2009년 동국대학교 대학원 정보통신공학과 수료(박사)
1995～2000 삼성SDI 중앙연구소 전임연구원
2000～2010 넷시큐어테크놀러지 부설연구소 연구기획팀장
2010～현재 KCC시큐리티 정보보호연구소 연구소장
관심분야 : 클라우드 분산 컴퓨팅, 정보보호, 네트워크 보안
E-mail : road2@kccsecurity.com

나 일 균

1985년 서강대학교 전자계산학과 졸업(학사)
1987년 서강대학교 전자계산학과 졸업(석사)
1994년 미국 콜로라도 대학교 대학원 (볼더 캠퍼스) 컴퓨터학과 졸업(석사)
2001년 미국 시라큐스 대학교 대학원 컴퓨터학과 졸업(박사)
2001～현재 미국 콜로라도 주립대학교 (덴버 캠퍼스) 컴퓨터공학과 부교수
관심분야 : 그리드/클라우드 분산 컴퓨팅, 네트워크 컴퓨팅
E-mail : ilkyeun.ra@ucdenver.edu

김 양 우

1984년 연세대학교 전자공학과 졸업(학사)
1986년 미국 시라큐스 대학교 대학원 컴퓨터공학과 졸업(석사)
1992년 미국 시라큐스 대학교 대학원 컴퓨터공학과 졸업(박사)
1996년～현재 동국대학교 정보통신공학과 교수
관심분야 : 그리드/클라우드 분산 컴퓨팅, 네트워크 컴퓨팅
E-mail : ywkim@dgu.edu