클라우드 스토리지 환경을 위한 안전하고 효율적인 암호데이터 중복제거 기술+

김원빈, 이임영
순천향대학교 컴퓨터소프트웨어공학과
e-mail:[wbkim29, imylee]@sch.ac.kr

A Safe and Efficient Secure Data Deduplication for Cloud Storage Environment

Won-Bin Kim, Im-Yeong Lee
Dept of Computer Software Engineering, Soonchunhyang University

요 약

기존의 암호데이터 중복제거 기술은 데이터의 중복 여부를 판단하기 위해 다양한 방식으로 데이터를 전송하고 이를 기존에 저장된 데이터와 비교하여 중복여부를 판단하게 된다. 이러한 데이터 중복제거 기술의 중복제거 효율성을 높이기 위해 최근 블록 단위의 중복제거 기술이 사용되고 있다. 하지만 블록 단위 중복제거 기술의 적용 과정에서 다양한 보안 위협이 발생하는데, 이 중 포이즌 어택은 무결성 및 데이터 저장 시 저장되는 데이터에 대한 검증이 이루어지지 않는 시스템에서 발생하는 위협 중 하나이다. 이러한 위협을 해결하기 위해 암호화 기술을 적용한 여러 기술들이 연구되어 제안되었지만 과도하게 많은 통신 횟수와 연산이 발생되어 효율성이 떨어지는 문제가 존재한다. 따라서 본 논문에서는 클라우드 스토리지에 저장되는 데이터의 기밀성과 무결성을 보장하며, 연산량과 통신량에서 보다 효율적인 암호데이터 중복제거 기술을 제안한다.

1. 서론

최근 통신기술의 발달과 통신기기들의 빠른 보급으로 매년 생성되는 데이터의 양은 2013년 4.4조 GB(Giga Bytes)에서 2020년 44조 GB까지의 증가가 예상될 정도로 갈수록 더 많은 데이터가 생성되고 있다. 매 시각 생성되는 수많은 데이터의 일부는 데이터 스토리지에 저장되어 보관된다. 개인 또는 기업에서 수없이 생성되는 데이터를 저장하기 위해서는 항상 가용용량이 확보되어야 하며, 데이터가 폐기되기 전까지는 해당 데이터를 안전하게 보관할 필요가 있다. 따라서 데이터의 양이 많아질수록 이를 유지하기 위한 비용이 커질 수밖에 없다. 이를 해결하기 위한 방안으로 같은 데이터를 한 번만 저장하여 스토리지의 저장 효율을 높이는 기술인 데이터 중복제거 기술이 연구되었다[1].

데이터 중복제거 기술은 다수의 사용자가 스토리지에 데이터를 저장하려 할 때 스토리지를 탐색하여 이미 저장되어있는 데이터인지를 파악하고, 이미 저장되어있는 데이터라면 해당 데이터의 주소 값만을 참조하는 방식이다. 이 방식은 같은 데이터를 여러 번 저장하지 않기 때문에 스토리지에 중복된 데이터가 저장되지 않아 저장효율을 높일 수 있다.

반면 데이터 중복제거의 가장 큰 문제점은 데이터의 기밀성 유지이다. 데이터 스토리지에 저장된 데이터는 공격에 의해 데이터가 탈취되었을 경우 기밀성을 유지해야할 필요가 있다. 이를 위해 저장된 데이터는 암호화가 적용되어야 한다. 데이터의 암호화가 적용될 경우 암호화 키에 따라 생성되는 암호문이 달라 암호화 된 데이터의 비교가 불가능하게 된다. 반면, 데이터 중복제거 기술은 두 데이터를 비교하여 같은지를 판단하는 기술이기 때문에 서로 상충되는 문제가 발생한다. 이러한 문제를 해결하기 위해 여러 기술들이 연구되었지만 기존에 연구된 기술들은 통신 횟수, 연산량이 비약적으로 많거나 보안위협에 취약한 문제가 존재한다. 따라서 본 제안방식에서는 효율성을 높이면서 보안위협을 고려한 암호데이터 중복제거 시스템을 제안한다.

2. 관련 연구

2.1 데이터 중복제거

다수가 사용하는 클라우드 스토리지는 영화, 음악, 응용프로그램 등 다양한 파일 저장된다. 이 중 대다수의 파일들은 기존에 저장된 파일과 동일하거나 일부만이 변형된 파일들로 구성되어있다. 이러한 파일들이 반복되어 저장될 때 파일의 데이터 중 동일한 부분을 반복하여 저장하지 않고 기존에 저장된 데이터를 참조하는 방식이 이용된다.

+ 본 논문은 중소기업청에서 지원하는 2014년도 산학연협력 기술개발사업(No. C0221609)의 연구수행으로 인한 결과물임을 밝힙니다.

이는 저장되는 데이터가 기존에 저장된 데이터와 일치할 때마다 저장 공간이 절약되는 효과를 준다. 따라서 데이터 스토리지의 저장 공간 효율을 향상시키기 위해 동일한 데이터를 한 번만 저장하는 기술인 데이터 중복제거 기술이 연구되고 있다.

2.2 중복제거 단위

중복제거의 단위는 크게 파일 단위 중복제거와 블록 단위 중복제거로 분류된다[2]. 파일 단위 중복제거 방식은 파일을 하나의 개체로 하여 비교하는 방식으로 빠른 중복여부 판단이 가능한 것이 특징이다. 하지만 파일의 데이터가 1bit만 달라져도 서로 다른 파일로 취급되어 중복제거가 이루어지지 않아 중복제거 효율이 낮은 단점이 있다.

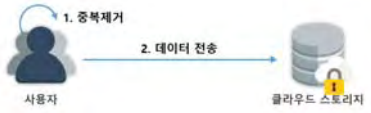
블록 단위 중복제거방식은 파일을 Chunk 또는 Segment라고 불리는 단위로 쪼개어 비교하는 방식이다. 데이터의 일부가 달라져도 달라진 데이터를 제외한 나머지 블록은 같은 데이터로 인식하여 중복제거가 이루어지게 된다. 따라서 파일 단위 중복제거 방식보다 처리 속도가 느리지만 중복제거 효율이 높은 장점이 있다.

2.3 중복제거 위치

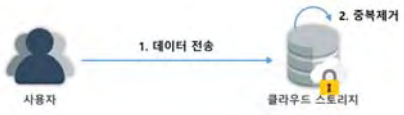
중복제거 기술은 중복제거를 수행하는 위치에 따라 크게 소스 방식과 타켓 방식으로 나뉜다[2]. 소스 방식은 사용자 기기에서 중복제거를 한 뒤 중복제거 된 데이터만을 클라우드 스토리지에 전송하는 방식으로 중복된 데이터는 클라우드 스토리지로 전송되지 않아 데이터 전송량이 비교적 낮다(그림 1). 반면 타겟 중복제거 방식은 클라우드 스토리지에서 중복제거를 수행하는 방식이다(그림 2). 중복된 데이터의 양과 관계없이 모든 데이터를 서버로 전송하기 때문에 소스 방식보다 비교적 많은 데이터 전송이 이루어진다. 하지만 일반적으로 사용자 단말기보다 서버의 하드웨어가 높은 성능을 가지고 있기 때문에 소스 방식보다 타겟 방식이 더 빠른 중복제거 속도를 보여주며, 사용자 단말기의 부하도 적게 발생한다.

2.4 중복제거의 문제점

일반적인 중복제거 기술은 암호화가 적용되지 않은 기술이다. 만약 데이터 스토리지에 저장된 데이터가 암호화되지 않으면 내·외부의 공격에 의해 데이터가 유출되었을



(그림 1) 소스 중복제거 방식



(그림 2) 타겟 중복제거 방식

(그림 3) Convergent Encryption 기술

경우 데이터의 기밀성을 보장하지 못한다. 따라서 데이터를 암호화하여 기밀성을 보장해야 하기 때문에 암호화를 적용한다. 암호화 기술은 서로 다른 사용자가 각자 자신만의 암호화 키로 동일한 데이터를 암호화 할 경우 서로 다른 암호문이 생성되기 때문에 복호화가 이루어지기 전까지는 두 암호데이터의 내용을 알 수 없게 하는 기술이다. 반면 중복제거라는 기술은 데이터의 비교를 통해 중복을 판단하는 기술이기 때문에 암호화된 데이터의 비교를 통해 중복제거를 할 수 없는 상황이 발생한다. 따라서 암호화된 데이터의 중복제거를 위해서는 특수한 방식의 암호화 기술이 적용되어야 한다.

2.5 암호화

암호화 알고리즘은 사용하는 환경과 목적에 따라 다양한 방식이 이용된다. 암호데이터 중복제거에서는 암호화를 적용하기 위해 중복제거에 필수적인 조건을 만족하는 알고리즘의 선택이 필요하다. 중복제거에 필수적으로 요구되는 암호화 알고리즘의 조건은 서로 다른 사용자가 같은 데이터를 암호화 할 경우 동일한 암호데이터가 생성되어야만 한다. 이를 위해 여러 가지 기술이 이용되지만 가장 대표적으로 이용되는 기술이 Convergent Encryption(CE, 수렴암호화)으로 원본 데이터의 해시값을 키로 이용하는 기술이다. 본 제안방식에서도 CE를 이용한 암호화를 진행한다(그림 3).

2.6 Wang. et al

Wang 등에 의해 제안된 기술로 Storer 등이 제안한 Secure Data Deduplication에 기반하여 설계되었다[3][4]. 이 기술은 클라우드 스토리지에 저장할 데이터의 메타데이터를 메타데이터 서버에 전송하여 중복여부를 판단한다. 이후 사용자는 중복되지 않은 데이터만을 클라우드 클라우드 스토리지에 전송하는 방식을 이용하기 때문에 사용자에서 중복제거를 하는 방식인 소스 중복제거 방식이 적용된다. 하지만 이 방식은 메타데이터 서버가 사용자에게 업로드 할 Chunk List 보내주지만 사용자가 스토리지 서버에 전송하는 Chunk들을 검증하는 과정이 없다. 이 때문에 사용자가 악의적으로 악성코드가 포함된 Chunk를 전송하여 저장될 경우, 추후 제 3자가 해당 Chunk를 내려받을 때 악성코드를 같이 내려받게 되는 문제가 발생할 수 있다.

2.7 포이즌 공격

메타데이터와 데이터 블록이 독립적으로 생성되어 따로 저장되며, 저장되는 두 데이터의 검증이 이루어지지 않을 경우 발생하는 위협이다. 데이터의 최초 업로더가 메타데이터 서버에는 원본 데이터의 업로드를 요청하고, 스토리지 서버에는 변형되거나 악성코드가 포함된 위·변조된 데이터를 업로드하는 공격이다. 이 위협은 제 3자가 원본 데이터의 다운로드를 요청할 경우 원본 데이터가 아닌 악성코드가 포함되거나 위·변조된 데이터가 다운로드 되어 제 3자에게 원본 손실이나 악성코드 피해가 발생하게 된다[5].

3. 보안 요구사항

본 장에서는 포이즌 공격을 방지하기 위해 다음과 같은 보안요소가 요구된다.

- 기밀성(Confidentiality) : 스토리지 서버에 업로드 된 데이터의 내용은 소유권이 없는 제 3자가 알 수 없어야 한다.
- 무결성(Integrity) : 전송 중 이거나 전송된 데이터는 내용이 변경되지 않아야 하며, 사용자가 전송하는 데이터의 목록은 메타데이터 서버의 중복제거 된 데이터을 기준으로 하여 변경되지 않아야 함
- 효율성(effectiveness) : 중복제거 기술의 저장공간 효율성을 보장하면서 중복제거 처리 과정에서의 연산량, 통신횟수에서와 보안 취약점에서의 효율성을 제공하여야 함

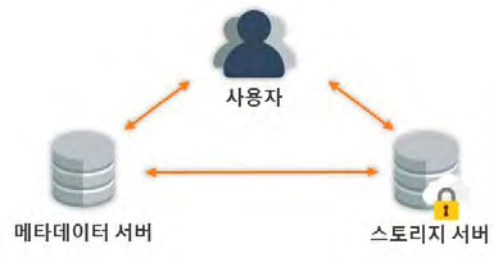
4. 제안방식

본 논문에서 제안하는 방식은 사용자, 메타데이터 서버, 데이터 스토리지로 구성되어 있다(그림 4). Wang 등이 제안방식에서 통신횟수를 그대로 유지하며 업로드 되는 검증하는 과정을 포함하여 포이즌 공격을 방지한다.

4.1 시스템 계수

본 제안방식에서는 다음과 같은 시스템 계수를 사용하여 프로토콜을 설계한다.

· * : 참여 객체(j : 사용자 j, M : 메타데이터 서버, SS : 스토리지 서버)
· f : 원본 파일



(그림 4) 본 제안방식의 구성 요소

· x : 중복된 $Chunk$의 수
· y : 중복되지 않은 $Chunk$의 수
· z : f로부터 생성된 모든 $Chunk$의 수
· k_{CE_Z} : $Chunk_z$를 암호화 할 CE 암호화 키
· C_z : 암호화키 k_{CE_Z}로 암호화 된 $Chunk_z$
· CID_z : $Chunk_z$의 식별자로 사용될 해시화 된 C_z
· CL : CID_z의 집합으로 이루어진 리스트
· CL_{Dedup} : 중복제거 된 CID_z의 집합
· $Dedup\{C_y\}$: 중복제거 된 C_z의 집합
· FID_f : 파일 f의 식별자로 사용될 해시화 된 f
· UID_j : 사용자 j의 식별자
· KL_f : 암호화 된 k_{CE_Z}의 집합
· $K_{S_{M,j}}$: 메타데이터 서버와 사용자 사이의 세션키
· K_* : *의 비밀키(대칭키)
· PK_* : *의 공개키
· SK_* : *의 개인키
· $H()$: 일방향 해시 함수

4.2 사용자

사용자는 원본 파일 f로 f의 식별자 FID_f, 사용자의 식별자 UID_j, 암호화된 $Chunk$ 리스트 CL, 암호화된 키 리스트 KL_f를 생성한다.

$$f = \{Chunk_0, Chunk_1, ..., Chunk_i\}$$
$$CL = \{CID_0, CID_1, \cdots, CID_z\}$$
$$KL_f = E_{k_j}(\{k_{CE_0}, k_{CE_1}, \cdots, k_{CE_z}\})$$

FID_f, UID_j, $E_{PK_M}(CL)$, KL_f를 메타데이터 서버로 전송하며, 메타데이터 서버로부터 CL_{Dedup}를 받아 CL_{Dedup}에 포함된 $Chunk$만을 스토리지 서버로 전송한다.

4.3 메타데이터 서버

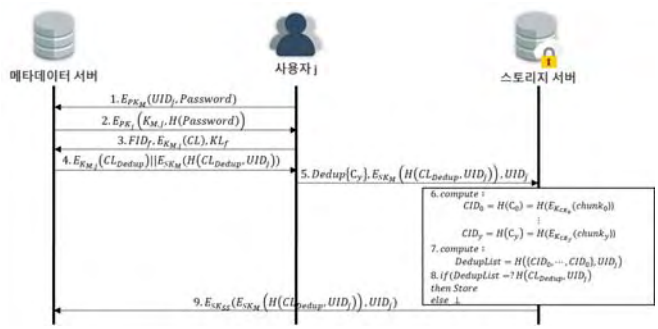
메타데이터 서버는 스토리지 서버에 저장된 $Chunk$의 메타데이터를 보관하며, 저장되어있는 메타데이터와 사용자가 보내준 FID_f, CL를 비교하여 저장된 데이터와의 중복여부를 판단한다. 사용자가 보내준 CL를 이용하여 중복되지 않는 데이터의 리스트 CL_{Dedup}을 생성하여 메타데이터 서버의 서명 $E_{SK_M}(H(CL_{Dedup}, UID_j))$과 함께 사용자에게 전송한다.

$$E_{SK_M}(H(CL_{Dedup}, UID_j))$$

메타데이터에는 원본 파일 f의 식별자 FID_f에 CL, KL_f, UID_j를 연결시킨 데이터로 구성되어 있다(그림 5).

File ID	Data		
FID_f	CL	KL_f	UID_j

(그림 5) 메타데이터의 구조



(그림 6) 제안 방식의 프로토콜

4.4 데이터 스토리지

데이터 스토리지는 사용자가 보내준 $Dedup\{C_y\}$와 $E_{SK_M}(H(CL_{Dedup}, UID_j))$를 비교하여 메타데이터 서버에서 생성한 CL_{Dedup}과 사용자가 보내준 $Dedup\{C_y\}$의 리스트가 같은지를 검증한다.

$$CL_{Dedup} = \{CID_y\}$$
$$Dedup\{C_y\} = \{C_y\}$$
$$CID_y = H(C_y) = H(E_{k_{CE_y}}(Chunk_y))$$

4.5 데이터 업로드 프로토콜

본 제안방식은 (그림 6)과 같이 총 6회의 통신으로 구성된다.

Step 1. 사용자 j는 메타데이터 서버의 공개키로 UID_j, $Password$를 암호화하여 메타데이터 서버로 전송한다.

Step 2. 메타데이터 서버는 UID_j와 $Password$를 이용하여 사용자 j를 확인하고, 세션키 $K_{M,j}$를 생성하여 해시화된 $Password$와 함께 사용자의 공개키로 암호화하여 사용자에게 전송한다.

Step 3. 사용자 j는 $Chunk_z$를 해시화 하여 k_{CE_z}를 생성한 뒤 k_{CE_z}로 $Chunk_z$를 암호화하여 C_z를 생성한다. 또한 C_z로 구성된 CL을 생성하고 k_{CE_i}의 목록을 사용자 j의 비밀키로 암호화 하여 KL_f를 생성한 뒤 FID_f, UID_j, $E_{K_{M,j}}(CL)$, KL_f를 메타데이터 서버로 전송한다.

Step 4. 메타데이터 서버는 $E_{K_{M,j}}(CL)$를 복호화하여 CL을 얻고 메타데이터 서버에 CL에 포함된 데이터와 동일한 데이터가 저장되어있는지를 판단한다. 저장이 되어있지 않은 CL의 데이터를 목록화하여 CL_{Dedup}을 생성한다. 그 다음 스토리지 서버에서 데이터 검증에 이용할 수 있도록 메타데이터 서버의 개인키로 서명 $E_{SK_M}(H(CL_{Dedup}, UID_j))$을 생성하고 사용자에게 전송한다.

Step 5. 사용자 j는 CL_{Dedup}을 확인하여 중복되지 않은 데이터 목록인 $Dedup\{C_y\}$를 생성한다. 그 다음 메타데이터 서버로부터 받은 서명값과 $Dedup\{C_y\}$, UID_j를 스토리지 서버로 전송한다.

Step 6, 7, 8. 스토리지 서버는 $Dedup\{C_y\}$를 해시화한 뒤 $E_{SK_M}(H(CL_{Dedup}, UID_j))$의 목록과 비교하여 사용자가 위·변조된 데이터의 전송여부를 검증한다.

Step 9. 스토리지 서버는 사용자로부터 받은 데이터의 검증 결과가 올바른 경우 자신의 비밀키로 메타데이터 서버의 서명값을 재서명하여 메타데이터 서버로 보낸다. 이를 통해 메타데이터 서버는 스토리지 서버에 정상적으로 데이터가 저장됐음을 확인하고 해당 데이터의 메타데이터를 저장한다.

5. 보안 요구사항 분석

• 기밀성(Confidentiality) : 스토리지 서버에 업로드 되는 데이터는 암호화를 적용하여 기밀성이 보장된다.

• 무결성(Integrity) : 전송되는 데이터는 암호화를 통해 통신 주체가 아닌 제 3자는 데이터를 변경할 수 없으며, 스토리지 서버에 업로드 요청된 데이터는 메타데이터 서버의 서명값과 비교를 통해 데이터의 무결성을 검증한다.

• 효율성(effectiveness) : 소스 기반의 중복제거를 통해 데이터 트래픽이 감소되고, 낮은 연산량과 통신횟수를 통해 효율성을 보장하면서도 무결성 검증을 통해 포이즌공격에도 안전하다.

6. 결론

최근 다양한 분야에서 클라우드 스토리지를 이용하고 있다. 최근 네트워크와 IT 기술의 발달로 PC 뿐 이를 해결한 본 제안방식을 이용하면 적은 통신횟수 및 연산량을 기대해 볼 수 있다. 또한 포이즌 공격과 같은 보안 위협에 안전하기 때문에 보다 안전한 환경에서 클라우드 시스템을 이용할 수 있을 것으로 판단한다.

참고문헌

[1] “디지털 유니버스 보고서(Digital Universe Study):대한민국 디지털 데이터 조사”, 한국EMC, 2014.

[2] 김건우, 주영현, 엄영익, “클라우드 스토리지 데이터 중복제거 기술 동향”, 한국통신학회 2012 년도 동계종합학술발표회, 2012.

[3] Can Wang, Zhi-guang Qin, Jing Peng, Juan Wangl, “A Novel Encryption Scheme for Data Deduplication System”, Communications, Circuits and Systems, 2010 International Conference on. IEEE, 2010.

[4] Storer, M. W., Greenan, K., Long, D. D., Miller, E. L. “Secure data deduplication” In Proceedings of the 4th ACM international workshop on Storage security and survivability. ACM. 2008.

[5] 박경수, 엄지은, 박정수, 이동훈, “안전하고 효율적인 클라이언트 사이드 중복 제거 기술”, 정보보호학회 논문지, 2015.