클라우드 스토리지 구조를 고려한 무결성 검증 기법 설계*

손정갑, 라쉬드 후세인, 이재경, 오희국
한양대학교 컴퓨터공학과
e-mail: jgson@infosec.hanyang.ac.kr

Design of Integrity Verification Scheme Based on Cloud Storage Architecture

Junggab Son, Rasheed Hussain, Jaekyung Lee, Heekuck Oh
Dept of Computer Science and Engineering, Hanyang University

요 약

클라우드 스토리지는 아직까지 충분한 안전성을 제공하지 못하기 때문에 완전히 신뢰할 수 없는 시스템으로 인식되고 있다. 클라이언트는 스토리지에 저장된 데이터의 손실을 우려하여 서비스의 사용을 꺼려할 수 있다. 반면에 클라우드는 손실이 발생했을 경우, 책임을 클라이언트에게 전가할 수 있다. 본 논문에서는 무결성 검증 기법을 설계하여 이러한 문제를 해결하고자 한다. 논문에서는 클라우드 스토리지 구조를 고려하여 설계하였으며, 공개적 검사, 데이터 업데이트, 스토리지 핵심기술 지원을 설계 목표로 한다. 설계한 무결설 검증기법을 통해 보다 안전한 클라우드 스토리지 시스템을 구축할 수 있다.

1. 서론

최근 클라우드 컴퓨팅이 발달함에 따라 이를 이용한 다양한 어플리케이션과 서비스들이 생겨나고 있다. 클라이언트는 자신의 컴퓨터나 다른 모바일 기기들에 데이터를 저장하고 프로그램을 설치하는 대신 클라우드가 제공하는 서비스들을 이용하고 클라우드 스토리지에 데이터를 저장한다. 또한, 기업들은 자체적인 전산 시스템을 구축하는 대신 클라우드 서비스 제공자와 계약하여 비용을 지불하고 필요한 만큼의 클라우드 컴퓨팅 서비스를 이용한다. 이러한 패러다임의 변화로 인해 클라이언트나 기업은 인프라 구축에 드는 비용뿐만 아니라 유지 및 보수에 드는 비용까지도 절약할 수 있다.

이와 같이 클라우드 컴퓨팅이 비용절감 및 이용의 편의성 측면에서 큰 장점을 지니고 있지만, 아직까지 클라우드 스토리지가 완전히 신뢰할 수 없다는 점은 단점으로 작용할 수 있다. 클라우드 컴퓨팅에서 가장 보안이 취약하며, 따라서 가장 큰 관심을 받고 있는 구성요소가 클라우드 스토리지이다. 클라우드 스토리지는 클라이언트의 중요한 데이터들을 저장하고 관리하므로 공격으로 저장된 데이터가 유출될 경우, 클라우드 서비스에 대한 신뢰도가 추락할 수 있다. 또한, 클라우드 스토리지는 다양한 오류 및 외부로부터의 공격에 의한 데이터 손실로부터 안전하지 못한 구조를 가지는데, 대표적인 예로 비잔틴 오류에 의한 데이터 손실을 들 수 있다 [1]. 여러 가지 이유로 인해 데이터가 손실되었을 때, 서비스 제공자는 클라우드 서비스의 신뢰도가 떨어지는 것을 막기 위해, 또는 보상을 피하기 위해 데이터가 손실되었다는 사실을 숨길 수 있다. 이 때, 만일 클라이언트가 자신의 데이터가 손실되었다는 것을 증명할 수 없다면, 그 피해는 전부 클라이언트가 부담하게 된다. 이러한 피해를 방지하기 위해서는 저장된 데이터에 대한 무결성을 검증할 수 있는 기법의 적용이 필요하다.

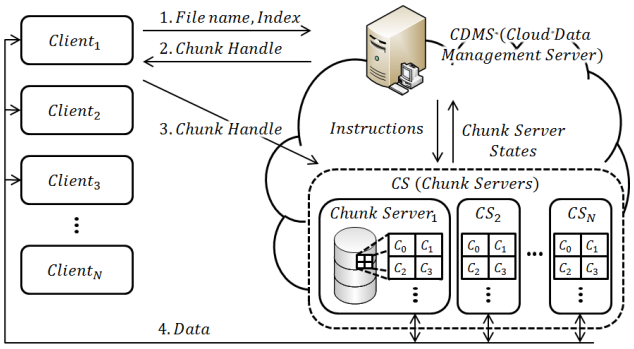
무결성 검증 기법에 대한 연구는 클라우드 스토리지의 단점을 극복하기 위한 기법으로 최근 관심을 가지며 많은 연구가 진행되었다[2-6]. 하지만, 선행 연구들은 challenge-response 방식으로 파일에 대한 주기적인 검사를 수행하도록 설계되었다. 이는 결국 추가적인 검사기관을 필요로 하게 되며, 잦은 검사로 클라우드에 많은 부담을 주게 된다. 또한, 클라우드 스토리지 구조를 고려하지 않고 설계되었으며, 따라서 이를 적용하기 위해서는 또 다른 문제들을 해결해야 한다.

본 논문에서는 클라이언트가 클라우드 서비스를 이용하다가 데이터가 손실되었을 때, 무결성 검증을 통해 손실이 클라우드 내부에서 발생하였음을 증명하는 기법을 설계한다. 제안하는 기법은 클라우드 레퍼런스 모델을 기반으로 설계하여 클라우드에게 추가적인 기능을 요구하지 않는

* 이 논문은 2013년도 정부(교육과학기술부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(No. 2012-R1A2A2A01046986).
* 이 논문은 2013년도 정부(교육과학기술부)의 재원으로 한국연구재단 기초연구사업의 지원을 받아 수행된 연구임 (No. 2012-R1A1A2009152)

장점이 있다. 또한, 클라우드 스토리지에 저장된 데이터를 공유하는 서비스가 많아짐에 따라 공개적인 검사가 가능하도록 설계하며, 데이터가 업데이트 되는 경우에도 효율적으로 이를 반영하도록 설계한다.

논문의 구성은 다음과 같다. 2장에서 클라우드 스토리지 구조에 대해 살펴보고 3장에서 설계 목표를, 4장에서



(그림 1) 클라우드 스토리지 레퍼런스 모델

배경지식에 대해 서술한다. 5장에서 제안하는 기법을 설계하며, 5장에서 결론을 맺는다.

2. 클라우드 스토리지 구조

SINA (Storage Networking Industry Association)의 Cloud storage reference model, GFS (Google File System) 등을 통해 클라우드 스토리지 구조를 알 수 있다. 클라우드 스토리지는 많은 기능을 제공한다. 하지만, 데이터 읽기, 쓰기의 단순한 기능만 고려한 스토리지 구조는 (그림 1)과 같이 표현할 수 있다. 클라우드 스토리지는 다수의 클러스터로 구성된다. 클러스터는 하나의 CDMS (Cloud Data Management Server)와 다수의 CS (Chunk Server), 다수의 클라이언트로 구성된다.

CDMS: 클러스터에 속한 사용자들의 전체 파일 시스템에 대한 metadata와 files directory structure를 관리한다. CDMS는 사용자가 파일을 요청할 경우, 이에 대한 metadata만 제공한다. CDMS는 CS에게 Chunk 생성, Chunk handle 할당 등의 명령을 전송한다. CS로부터 주기적으로 상태정보를 전송받는다.

CS: CS는 고정 크기의 Chunk들로 구성된다. GFS의 default chunk size는 64M이다. CS는 CDMS의 요청에 의해 Chunk를 생성하며, CDMS는 chunk에게 고유의 64bit chunk handle을 할당한다. 클라이언트의 데이터는 손실과 성능 향상을 위해 여러 CS에 중복 저장된다. 중복의 레벨은 사용자의 요청 또는 서비스 정책에 따라 달라질 수 있다.

Client: 클라이언트는 자신이 속해 있는 클러스터 내의 CDMS와 통신한다. CDMS로부터 File directory structure를 전송받으며 읽기, 쓰기 등의 연산을 요청한다. 데이터를 요청하면 CDMS는 데이터의 metadata를 클라이언트에게 전송한다. 클라이언트는 이 metadata를 CS에게 전달하여 필요한 데이터를 전송받는다.

3. 설계 목표

제안하는 기법은 클라우드 서비스 제공자가 데이터 손실을 숨길 수 없도록 무결성 검증기법을 설계하는 것을 목표로 한다. 기존 challenge-response 방식으로 설계하면 클라이언트의 부담은 줄일 수 있지만 response를 생성하는 과정에서 CDMS와 CS 내부 연산은 오히려 늘어나게 된다. 또한, response를 계산하는 과정이 간단하지 않으므로 제 3자가 주기적으로 검사를 수행한다는 가정 자체가 클라우드 스토리지에 엄청난 오버헤드를 주게 된다. 데이터를 가지고 검증과정을 수행하게 되면 검증과정을 보다 간단하게 설계할 수 있으며, 스토리지의 부담을 덜어줄 수 있다. 제안하는 기법은 클라우드 스토리지 구조를 기반으로 블록 기반 검증기법을 설계하며 추가로 다음과 같은 특징을 가진다.

3.1 공개적 검사

최근 클라우드 컴퓨팅 서비스는 단일 서비스 뿐만 아니라 다른 서비스를 위한 매개체 역할을 하는 경우가 많다. 제안하는 기법은 이러한 점을 고려하여 데이터 소유자 뿐만 아니라 접근 가능한 모든 사용자가 무결성을 검사할 수 있도록 설계한다. 그러므로 무결성 검증기법은 공개키를 기반으로 설계되어야 한다.

3.2 데이터 업데이트

클라우드 스토리지에 저장된 데이터는 다양한 어플리케이션을 통해 수정될 수 있다. 무결성 검증 기법이 이를 고려하지 않으면, 데이터의 일부가 수정되더라도 전체 데이터에 대한 증거를 다시 생성해야 한다. 제안하는 기법에서는 블록 단위로 설계하여 증거의 업데이트를 반영할 수 있도록 설계한다.

3.3 스토리지의 핵심 기술 지원

클라우드 스토리지는 성능과 신뢰성 향상을 위해서 멀티테넌시와 중복 제거 기술을 사용한다. 무결성 검증기법이 이를 고려하지 않고 설계되면 스토리지의 성능을 감소시킬 수 있으므로 이러한 기술을 그대로 적용할 수 있도록 설계하여야 한다.

4. 표기법과 배경 지식

4.1 표기법

<표 1>은 논문에서 사용하는 표기법을 나타낸다.

4.2 Bilinear map

논문에서는 데이터 무결성 검증 단계에서 bilinear map을 사용한다. 다음과 같은 특성을 만족하는 $e: G \times G \rightarrow G_T$를 bilinear map이라 한다[7].

- Bilinear:

$\forall h_1, h_2 \in G$ and $a, b \in Z_q, e(h_1^a, h_2^b) = e(h_1, h_2)^{ab}$

- non-degenerate: G의 모든 쌍 P, Q에 대하여 $e(P,Q)$는 G_T의 항등원이 아니어야 한다.

- Computable: 임의의 P, Q에 대하여 $e(P,Q)$를 계산할 수 있는 효율적인 알고리즘이 존재해야 한다.

<표 1> 표기법

Notation	Description
q	k-bit prime number
Z_q	Integers modulo q
G, G_T	Cyclic group of prime order q
e	Bilinear pairing that satisfies with $G \times G \rightarrow G_T$
pk, sk	Public, private key pair
M	Set of data blocks $\{m_1, m_2, m_3, \ldots, m_n\}, m_i \in Z_q$
P	Proof
ρ	Set of proof of updated data, $\{P_1, P_2, P_3, \ldots, P_j\}$
g	Generator of G
h_1	$0,1^* \rightarrow G$
h_2	$0,1^* \rightarrow Z_q$

5. 무결성 검증 기법 설계

본 절에서는 클라우드 스토리지 레퍼런스 모델을 기반으로 저장된 데이터의 무결성을 보장하는 기법을 설계한다. 무결성 검증 기법은 클라이언트뿐만 아니라 클라우드 스토리지도 많은 오버헤드를 주지 않도록 설계한다.

클라우드 스토리지에 클라이언트의 데이터는 블록 단위로 저장되며 파일 디렉토리 구조는 CDMS에 의해 관리된다. 그러므로 CS에는 각 블록에 대한 증거를 저장하고 CDMS에는 전체 파일에 대한 증거를 저장한다. 이러한 두 가지 증거를 사용함으로써 증거 자체에 대한 손실을 확인할 수 있으며, 데이터의 업데이트에 따른 증거의 업데이트를 효율적으로 수행할 수 있다.

5.1 설정

제안하는 기법에서는 파일 M이 n개의 블록, $\{m_1, m_2, m_3, \ldots, m_n\}, m_i \in Z_q$ 로 구성되어 있다고 가정한다. 클라이언트들은 스토리지에 데이터를 저장하기 전에 KeyGen알고리즘을 통해 (pk, sk)를 생성하거나 TTP(Trust Third Party)가 생성한 (pk, sk)를 전달받는다.

KeyGen(1^k): 클라이언트의 키 쌍을 생성하기 위해 $x \in_R Z_q$를 선택하고 공개키와 개인키를 각각 $pk = g^{sk}, sk = x$로 설정한다.

5.2 데이터 저장

클라이언트는 파일 M을 저장하기 위해 증거 P를 생성한다. 주어진 $M = (m_1, m_2, m_3, \ldots, m_n)$ 에 대하여 $u \in_R G$를 생성하고 다음을 계산한다.

$$H = \prod_{i=1}^{n} h(m_i) \in G, r = \sum_{i=1}^{n} h(m_i) \in Z_q \quad (1)$$

P는 $H \bullet u^r$을 sk로 서명한 값이 된다. $P = (H \bullet u^r)^{sk}$

클라이언트는 M과 P를 CDMS로 전송한다. CDMS는 데이터의 무결성을 확인한 후, 이를 스토리지에 저장한다.

M에 대한 무결성을 검증하기 위해 식(1)을 이용하여 $H \bullet u^r$을 생성한다. 생성한 값과 pk를 페어링 값이 P와 g의 페어링 값과 같을 경우, TRUE를 출력한다. 아닐 경우, FALSE를 출력한다.

If $e(H \bullet u^r, g^{sk}) = e(P, g)$ then TRUE, otherwise $\perp$.

5.3 데이터 업데이트

데이터의 업데이트는 이미 데이터가 저장되어 있는 상태에서만 진행된다. 저장된 데이터가 업데이트될 경우, 데이터뿐만 아니라 P도 역시 업데이트되어야 한다. 데이터의 업데이트가 발생하는 동적환경을 효과적으로 지원하기 위해서 batch verification을 적용한다. 업데이트 된 데이터에 대한 증거들을 스토리지에 저장해 두었다가 무결성 검증이 필요한 순간에 이를 결합하여 검증 절차를 진행한다. 데이터는 블록 단위로 업데이트되며 삽입, 삭제, 수정이 발생한다고 가정한다. 클라이언트는 DataUpdate를 실행하여 업데이트되는 블록에 대한증거를 생성한다. m_i'는 업데이트 될 데이터 블록을 나타낸다.

*DataUpdate*는 *operation*값에 따라 다른 절차를 수행한다. *operation*이 *insert*일 경우, $H_i' = h(m_i') \bullet u^{h(m_i')}$를 생성하고 sk로 서명한 후 다음을 출력한다. $P_i = H_i'^{sk}$.

*operation*이 delete일 경우, $H_i' = h(m_i') \bullet u^{h(m_i')}$를 생성하고 sk로 서명한 후 다음을 출력한다. $P_i = (1/H_i')^{sk}$.

마지막으로 *operation*이 modify일 경우, 이전 블록인 m_i에 대해 $H_i = h(m_i) \bullet u^{h(m_i)}$를 계산하고, 수정될 블록인 m_i'에 대해 $H_i' = h(m_i') \bullet u^{h(m_i')}$를 계산한다. H_i와 H_i'을 sk로 서명한 후 다음을 출력한다. $P_i = (H_i'/H_i)^{sk}$.

저장된 블록에 대한 증거 P_i를 생성하여 스토리지에 전송하면, 스토리지는 $VerifyProof(M, P, pk)$를 실행하여 결과값이 *TRUE*일 경우, 업데이트를 반영한다.

5.4 업데이트된 데이터의 검증

업데이트된 데이터에 대한 증거는 기존 증거와 같이 스토리지에 저장된다. 데이터가 j번 업데이트되었다고 가정하면 $\rho = P_1, P_2, P_3, \ldots, P_j$ 가 생성된다. 업데이트된 데이터에 대한 무결성 검증이 필요할 경우, *VerifyUpdate*를 수행한다.

업데이트된 M'을 검증하기 위해 기존 저장된 P와 업데이트사항에 대한 검증 ρ를 결합한다.

$$P \bullet \rho = H^{sk} \bullet H_1^{sk} \bullet H_2^{sk} \cdots H_j^{sk} \bullet u^{r + m_1' + m_2' + \cdots + m_j'} \in G$$

식 (1)을 이용하여 M'에 대한 해쉬값 $H' \bullet u^{r'}$을 계산한다. 생성한 $H' \bullet u^{r'}$과 pk를 페어링한 값이 $P \bullet \rho$와 g

의 페어링 값과 같을 경우, *TRUE*를 출력한다. 아닐 경우, *FALSE*를 출력한다.

If $e(H' \cdot u^{r'}, g^{sk}) = e(P \cdot \rho, g)$ then TRUE, otherwise ⊥.

저장된 데이터의 업데이트가 빈번하게 발생하여 ρ의 길이가 길어질 경우, 서비스제공자는 페어링 연산의 homomorphic 특성을 이용하여 이를 줄일 수 있다. 그 $u_1, u_2, v \in G$에 대해 $e(u_1 u_2, v) = e(u_1, v) \cdot e(u_2, v)$를 만족하는 bilinear map[7]을 이용하면, 다음과 같은 응용이 가능하다. $e(P,g)$를 미리 계산하여 저장해 놓고, 나중에 발생한 ρ에 대해 homomorphic 특성을 이용하여 다음과 같이 무결성을 검증할 수 있다.

$$\begin{aligned} e(H' \cdot u^{r'}, g^{sk}) &= e(P,g) \cdot e(\rho,g) \\ &= e(P \cdot \rho, g) \end{aligned}$$

6. 결론

본 논문에서는 클라우드 스토리지가 완전히 신뢰할 수 없다는 단점을 보완하고 클라이언트의 데이터를 보호하기 위해 무결성 검증 기법을 설계하였다. 무결성 검증 기법은 클라우드 스토리지 구조를 고려하여 설계하였으며, 공개적 검사, 데이터 업데이트, 스토리지 핵심기술 기능을 제공할 수 있도록 설계되었다. 이러한 기능들은 CDMS는 파일 레벨의 증거를, CS에는 블록 레벨의 증거를 저장함으로써 제공 가능하다. 논문에서 설계한 무결설 검증기법을 통해 보다 안전한 클라우드 스토리지 시스템을 구축할 수 있다.

참고문헌

[1] S. Bonomi, and A. S. Nezhad, "Multi-Writer Regular Registers in Dynamic Distributed Systems with Bizantine Failures," Proceedings of the 3rd International Workshop on Theoretical Aspects of Dynamic Distributed Systems (TADDS '11), pp. 8-12, 2011.

[2] K. Yang, X. Jia, "Data Storage Auditing Service in cloud computing: Challenges, Methods and Opportunities," Journal World Wide Web, Vol. 15, Issue. 4, 2012.

[3] C. Erway, A. Kupcu, C. Papamanthou, and R. Tamassia, "Dynamic Provable Data Possession," Proceedings of the 16th ACM conference on Computer and Communications Security (CCS '09), 2009.

[4] D. Boneh, B. Lynn, and H. Shacham, "Short Signatures from the Weil Pairing," Proceedings of 7th International Conference on Theory and Application of Cryptology and Information Security: Advances in Cryptology (ASIACRYPT '01), pp. 514-532, 2001.

[5] Q. Wang, C. Wang, K. Ren, W. Lou, and J. Li, "Enabling Public Auditability and Data Dynamics for Storage Security in Cloud Computing," IEEE Transactions on Parallel and Distributed Systems, Vol. 22, No. 5, pp.847-859, May 2011.

[6] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song, "Provable Data Possession at Untrusted Stores," Proceedings of the 14th ACM conference on Computer and Communications Security (CCS '07), pp. 598-609, 2007.

[7] D. Boneh, B. Lynn, and H. Shacham, "Short Signatures from the Weil Pairing," Proceedings of 7th International Conference on Theory and Application of Cryptology and Information Security: Advances in Cryptology (ASIACRYPT '01), pp. 514-532, 2001.