공개검증 기법에 대한 연구*

은하수†, 이훈정, 이재경, 오희국‡
한양대학교 컴퓨터공학과
e-mail : hseun@infosec.hanyang.ac.kr

A Study on Scheme for Public Auditing*

Hasoo Eun†, Hoonjung Lee, Jaekyung Lee, Heekuck Oh‡
Dept. of Computer Science and Engineering, Hanyang University

요 약

클라우드 컴퓨팅의 출현은 외부 스토리지(예를 들면, 클라우드 스토리지)에 저장되는 사용자의 데이터를 크게 증가시켰다. 사용자는 자신의 데이터가 저장되어 있음을 알 수 있지만, 어떻게 관리되는지 알 수 없다. 이러한 상황에서 서버 오류나 해커의 공격 등을 통해 데이터 손실이 발생하게 된다면, 그 피해는 고스란히 사용자에게 돌아가게 된다. 만일 피해를 입은 데이터가 잘 사용되지 않는 데이터라면 서버는 오류 사실을 은닉하고 정상적인 데이터처럼 보이거나, 해당 데이터를 소거해 버릴 수도 있다. 따라서 사용자는 자신의 데이터를 보호하기 위해 외부 스토리지에 저장된 데이터를 검증할 필요가 있다. 본 논문에서는 클라우드 컴퓨팅이 등장하기 이전에 제안된 기법들에서부터 최근 제안된 기법들까지 정리하고 이들에 대한 모델을 세워 분류한다. 또한 각 모델에서 발생할 수 있는 문제점들을 분석하여 보안 요구사항을 정리한다.

1. 서론

데이터를 외부 스토리지에 저장하는 것은 접근성을 높이고, 데이터를 공유하는 방법의 하나로써 널리 사용되고 있다. 클라우드 컴퓨팅의 등장은 사용자의 외부 스토리지(예를 들면, 클라우드 스토리지) 사용을 더욱 가속화시켰다. 현재 Naver nDrive, Daum Cloud, Google Drive 등 다양한 클라우드 스토리지 서비스가 제공되고 있고, 많은 사용자들이 이용하고 있다. 사용자들은 자신의 데이터를 클라우드에 저장하고 다양한 기기를 통해 이에 접근할 수 있다. 하지만 이것이 어떻게 저장되어 있고, 관리되는지는 알 수 없다.

사용자가 데이터를 저장한 클라우드 스토리지에 해킹이나 비잔틴 오류가 발생하여 데이터 손실이 발생했다고 가정해보자. 만일 사용자가 데이터 사본을 유지하고 있지 않다면, 이는 곧 사용자 데이터 손실로 연결된다. 손실된 데이터가 가끔 사용되는 것이라면, 스토리지 서비스 제공자는 데이터를 소거하거나 혹은 다른 방법으로 사건을 감추려 할 수 있다. 따라서 사용자는 자신의 데이터를 보호하기 위해, 저장된 데이터가 올바르게 유지되고 있는지 확인할 수 있어야 한다.

데이터 변경여부를 확인할 수 있는 가장 쉬운 방법은 해쉬(Hash)를 이용하는 것이다. 해쉬는 입력 값에 미세한 변화가 생겨도 결과에 큰 영향이 생기는 성질(눈사태 효과, Avalanche effect)을 가지고 있기 때문에 데이터의 해쉬 값을 유지하기만 해도 쉽게 자신의 데이터가 정상적으로 저장되어 있는지 확인할 수 있다. 하지만 해쉬 값만 사용하게 되면 스토리지 서비스 제공자의 재전송 공격에 대응할 수 없다. 이를 해결하기 위해 준동형 서명에 기반을 둔 기법들이 제안되기 시작했다.

스토리지에 데이터가 온전히 저장되어 있는지 확인하기 위해서는 검증자가 검증을 위한 값을 유지하고 있어야 한다. 초기의 연구들은 데이터를 저장한 사용자가 검증을 수행했다. 따라서 검증에 사용되는 값을 사용자가 유지했다. 이 경우 검증자로 인한 평문 노출 등의 부담이 없었기 때문에 검증 값을 생성하기 위해 평문의 일부를 사용하는 등 다양한 기법이 제안되었다. 하지만 검증 값을 유지하는 비용과 검증을 수행하기 위한 통신 비용이 사용자에게 부담된다는 의견과 함께 제 3 검증자 (TPA, Third Party Auditor)를 두어 검증을 수행하는 기법들이 제안되었다. 제 3 검증자를 이용하면 검증과정에 사용자의 부담이 없어진다. 하지만 이를 위해 스토리지 서비스 제공자와 제 3 검증자 사이에 공모가 없어야 하며, 제 3 검증자에게 사용자의 데이터가 노출되지 않도록 보호해야 한다.

본 논문에서는 앞서 언급한 공개검증 기법을 흐름에 따라 살펴보고 이들을 분류하여 모델을 수립한다. 이후 각 모델에 따라 고려되었던 사항과 연구의 흐름을 정리하여, 향후 공개검증 기법의 연구 방향을 모색한다.

본 논문의 구성은 2 장에서 공개검증 기법과 관련

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

된 기초지식들에 대해 설명하고 3 장에서 현재까지 제안된 공개검증 기법을 소개한다. 4 장에서는 3 장에서 소개한 기법들에 따라 모델을 세우고 분류하고, 분석한다. 5 장에서는 4 장의 분석 결과를 토대로 향후 공개 검증 기법의 연구 방향을 모색하고 결론을 맺는다.

2. 기초지식

본 장에서는 공개검증 기법들에서 일반적으로 사용되는 환경 및 기초지식에 대해 소개한다. 공개검증 기법 연구가 외부 스토리지 검증에서 유래된 만큼, 거의 동일한 환경을 띄고 있다. 본 논문에서는 외부 스토리지 검증 기법을 공개검증 기법의 초기 연구로 판단하고 이를 혼용하여 사용한다.

가. 참여자

공개검증 기법의 참여자는 크게 소유자, 서비스 제공자, 검증자로 구분할 수 있다. 소유자는 데이터에 대한 소유권을 가진 사용자를 말한다. 소유자는 자신의 데이터를 서비스 제공자에게 맡기고, 데이터가 잘 유지되는지 확인하고자 한다. 서비스 제공자는 소유자의 데이터를 저장하는 개체를 말한다. 실제 저장은 스토리지에 되고 서비스 제공자는 관리를 하는 주체이지만, 본 논문에서는 이를 같은 의미로 혼용하여 사용한다. 검증자는 스토리지에 저장된 소유자의 데이터를 검증하는 개체를 말한다. 검증자는 소유자 본인이 될 수도 있고, 제 3 자가 될 수도 있다.

나. 로컬 데이터 검증과의 차이점

공개검증은 로컬 데이터 검증과 달리 추가적으로 고려해야 할 사항이 있다. 이는 두 검증 시스템의 환경적 차이로부터 시작된다. 본 연구의 분석 결과 공개검증 시 추가적으로 고려해야 할 사항은 다음과 같다.

로컬 데이터 사본 없음: 로컬 데이터 사본은 소유자가 외부 스토리지에 저장한 데이터 외에 로컬 스토리지에 동일 데이터를 유지하는 것을 말한다. 일반적인 공개검증 기법에서 로컬 데이터 사본이 없다고 가정하고 있다. 소유자가 외부 스토리지 서비스를 이용함에 있어 로컬 데이터 유지에 필요한 비용을 줄이기 위해서라는 의견이다. 이를 위해 시나리오 상의 소유자는 외부 스토리지에 데이터를 저장하고 단말에 남은 데이터를 지운다.

대용량 데이터에 대한 다수 사용자의 접근: 공개검증 기법에 사용되는 데이터는 용량이 크고, 여러 사람이 함께 사용하는 것을 목표로 한다. 한 예로 천문학 데이터를 들 수 있다. 이러한 데이터의 경우 여러 사람이 접근해야 하고 사용해야 하기 때문에, 데이터 사용자들이 검증을 수행할 수 있거나 검증의 결과를 알 수 있어야 한다.

검증에 필요한 통신 비용: 로컬 데이터 검증과 달리 공개검증 기법은 온라인으로 검증에 필요한 데이터를 주고 받는다. 검증을 위해 전체 데이터를 이용할 수 있지만 I/O 및 통신 비용이 너무 크기 때문에 이러한 방법은 지양해야 한다. 검증에 사용되는 데이터는 작을수록 좋다.

제 3 검증자로부터 데이터 보호: 최근에 제안되고 있는 기법들은 검증만 수행하는 개체를 두어 검증에 필요한 소유자의 오버헤드를 줄이고 있다. 하지만 검증자가 평문 데이터를 사용하면, 검증자를 통해 데이터 유출이 발생할 수 있다. 이는 사용자의 프라이버시, 나아가 회사의 이익에도 영향을 줄 수 있다. 제 3 검증자는 검증을 수행하되 신뢰하지 않는 기관이라 가정한다.

3. 공개검증 기법

본 장에서는 초기에 제안되었던 무결성 검증 기법에서부터 최근 제안되고 있는 공개검증 기법까지 소개한다. 본 논문에서는 기존의 기법들을 해쉬기반 검증 기법, 준동형 태그기반 검증 기법, 제 3 검증기관을 통한 검증 기법 등 셋으로 분류하였다.

가. 해쉬기반 검증 기법

외부 스토리지 검증기법의 초기 연구 대부분 해쉬(또는 MAC)에 기반을 두고 있다. 그만큼 해쉬는 무결성 검증에 일반적으로 사용되는 기법이다. 해쉬 기반의 검증 기법은 서버가 자신이 저장한 데이터의 해쉬 값을 보냈을 때, 검증자가 유지하고 있는 해쉬 값과 일치하는지 확인하는 형태로 진행된다. Lillibridge 등은 리드-솔로몬 삭제 수정 코드(Reed-Solomon erasure-correcting code)를 여분 블록에 삽입하는 기법을 제안했고[1], Naor 등은 오류 수정 코드(Error-correcting code)를 파일과 블록에 삽입하고 해쉬를 하여 전달하는 기법을 제안했다[2]. 두 기법 모두 데이터 손실 여부를 확인할 수 있지만 원본 데이터를 사용자에게 다시 보내야 하기 때문에 통신비용이 크다는 단점이 있다. Jules 등은 파일을 블록으로 나누고, 각 블록을 오류 수정 코드로 인코딩하는 기법을 제안했다[3]. 암호화된 파일 사이에 검증을 위한 블록(Sentinel)을 함께 보내고, Sentinel 을 통해 검증을 수행한다. 이 기법은 서버가 Sentinel 을 구분해낼 수 없기 때문에 안전하게 보일 수 있지만, Sentinel 의 수에 따라 횟수가 제한된다는 문제가 있다.

해쉬기반 기법은 연산이 가볍다는 큰 장점을 가지고 있지만, 반면 원본 데이터가 없이 새로운 해쉬 값을 만들어 낼 수 없다는 한계가 있다. 이에 대해 Shah 등은 랜덤 값을 이용해 여러 개의 해쉬 값을 만들고 한 번에 하나씩 해쉬 값을 사용하며 검증하는 기법을 제안했다[4]. 하지만 질의 횟수가 정해져 있고, 검증을 수행하는 개체는 모든 파일에 대해 여러 개의 해쉬 값을 유지해야 하는 부담이 있다.

나. 준동형 태그기반 검증 기법

준동형이란 $f:\mathbb{P}\rightarrow\mathbb{Q}$ 인 함수가 있을 때 $f(g_1\oplus g_2)=f(g_1)\otimes f(g_2)$ 와 같은 성질을 말한다. 이때 $g_1,g_2\in\mathbb{P}$ 이고, $\oplus$ 와 $\otimes$ 는 각각 $\mathbb{P}$ 와 $\mathbb{Q}$ 의 연산이어야 한다. 이와 같은 특성은 지수연산에서 쉽게 찾을 수 있다.

연구의 흐름으로 보았을 때 준동형 태그의 등장은

질의 횟수가 한정적인 해쉬기반 기법을 개선하기 위해 제안된 것으로 판단된다. 준동형 태그를 이용하면 원본 데이터가 없어도 검증을 위해 필요한 데이터를 생성할 수 있다.

우리가 알기로 Deswarte 등은 최초의 준동형 태그 기반 공개검증 기법을 제안했다[5]. 제안된 기법은 서버가 생성한 값과 사용자가 생성한 값을 결합하여 검증하는 것으로서, 마치 디피-헬만 키교환 기법과 닮아 있다. 사용자는 데이터 업로드 시 유지하고 있던 $t = a^m \bmod N$ 와 $R = a^r \bmod N$ 을 서버에게 전송한다. 서버는 $P = R^m \bmod N$ 을 사용자에게 전송하고 사용자는 $P = ?t^r$ 을 통해 데이터를 소유하고 있는지 확인할 수 있다.

이 기법은 서버가 생성하는 검증 값에 사용자의 질의가 반영된다는 장점이 있다. 또한 파일을 전부 유지하지 않고도 횟수에 제한 받지 않고 질의 할 수 있다. 하지만 데이터 전체를 지수에 취해야 하기 때문에 사용자와 서버 입장에서 연산 비용이 크다는 단점이 있다.

Filho 등은 오일러 함수를 해쉬처럼 사용하여 서버의 데이터를 검증하는 기법을 제안했다[6]. 이 기법은 Deswarte 등의 기법에서 사용자가 유지하는 값의 크기와 질의를 위해 성성하는 연산을 줄인 기법이다. Deswarte 등의 기법보다 효율적이기는 하지만 여전히 파일 전체를 지수에 취해야 하기 때문에 연산에 대한 부담이 크다.

Yamamoto 등 메시지를 메시지를 블록으로 나누고, 각 블록마다 사용자의 챌린지와 결합하여 증거를 생성하는 기법을 제안했다[7]. Ateniese 등은 이를 몇몇 데이터 블록만 선택적으로 검사(샘플링)할 수 있도록 개선하였다[8]. Yamamoto 등의 기법은 전체 데이터를 사용하기 때문에 한번에 완전히 검사할 수 있지만, 샘플링 하는 것에 비해 연산량이 많다. 반면 샘플링을 하게 되면 적은 데이터로 검사를 할 수 있기 때문에 효율적이기는 하지만, 오류가 발생한 블록에 샘플에 포함되지 않을 수 있다는 문제가 있다. 최근 샘플링을 이용하는 기법들은 샘플링하여 반복적으로 수행하여 확률을 높이는 방법을 택하고 있다.

다. 제 3 검증기관을 통한 검증 기법

앞서 언급한 기법들은 데이터 소유자(클라이언트)가 검증을 수행했다. 하지만 최근의 기법들은 이를 제 3 검증기관에 위탁하고 검증의 결과만 사용자에게 전달하는 형태로 바뀌고 있다. 이와 같은 변화는 '키의 소유자' 또는 '데이터 소유자' 등의 제한에서 벗어나, 여러 사용자에게 균일한 수준의 검증 레포트를 제공해줄 수 있다는 장점이 있다. 또한 최근 사용자들이 모바일 단말을 주로 사용하기 때문에 기존 환경과 같이 직접 검증을 수행하게 되면 연산량뿐만 아니라 베터리, 통신량 등 더 많은 오버해드가 발생할 수 있다. 하지만 제 3 검증기관을 통하게 되면 사용자의 데이터가 원치 않게 노출될 수 있으므로 이를 보호하기 위한 기법들이 함께 제안되고 있다. 이에 Q. Wang 등은 기존의 기법을 제 3 검증기관에서 수행할 수 있게 하는 한편, 사용자 데이터의 동적 업데이트를 지원하기 위해 머클 해쉬 트리(Merkle Hash Tree)를 이용한 기법을 제안하였다[9].

4. 각 검증 모델에 대한 분석

본 장에서는 앞서 소개한 세가지 검증 모델에 대해 분석하고 장단점에 대해 서술한다.

해쉬기반 기법은 사용자가 파일에 대한 해쉬 값을 유지하고 서비스 제공자로부터 전달 받은 해쉬 값과 비교를 하는 형태로 진행된다. 연산량 측면에서 가장 저렴하지만, 원본이 없이는 새로운 질의를 생성할 수 없기 때문에 재전송 공격에 취약하다. 재전송 공격을 막기 위한 여러 기법들이 제안되었으나 질의 횟수에 제한을 받는 한계가 있다. 하지만 사용자, 서비스 제공자, 검증자 모두에게 가장 부담이 적기 때문에 검증 기간과 검증 횟수를 조절하면 가장 현실적인 대안이 될 수 있다.

준동형 태그기반 기법은 해쉬기반 기법의 검증 횟수 제한을 해결하고 사용자의 질의를 반영한 이상적인 기법이다. 하지만 관련 기법들의 연산량이 높기 때문에 사용자, 서비스 제공자, 검증자 모두에게 부담이 된다. 이를 해결하기 위해 데이터를 블록단위로 나누고 이를 다시 섹터단위로 나누어 사용자의 챌린지와 결합하는 기법들이 제안되었다. 이와 함께 선택적으로 몇몇 블록들만 검사하는 샘플링 기법이 제안되었지만, 비결정적 기법이다 보니 블록을 바꾸어가며 반복수행으로 확률을 높이고 있다.

제 3 검증기관을 통한 기법은 앞서 설명한 두 기법과 별도로 선택적 사용이 가능하다. 즉, 제 3 검증기관을 두지 않고 사용자가 직접 검증을 수행할 수도 있다. 제 3 검증기관이 참여하면 사용자가 검증을 수행하지 않기 때문에 사용자의 부담이 크게 줄어든다. 하지만 데이터가 제 3 검증기관에게 노출되지 않도록 하기 위해 부가적인 연산이 필요하기 때문에 검증에 필요한 연산량은 오히려 증가하게 된다.

5. 결론 및 향후 연구

본 논문에서는 공개검증 기법에 대한 연구들을 세가지 모델로 분류하여 소개하였고, 이들의 장단점과 해결해야할 과제에 대해 서술하였다. 공개검증 기법의 핵심은 서비스 제공자가 검증 값 생성 시 원본 데이터를 사용하게 만드는데 있다. 초기의 연구는 이를 위해 원본 데이터를 통째로 사용하였고, 최근에는 분할하고 선택적으로 사용하고 있다. 물론 앞으로의 연구도 이에서 크게 벗어나지 않는 형태를 취할 것으로 예상된다. 최근 제안된 Wang 등의 기법은 기존 기법들의 총집합이라 할 만큼 기존 연구들을 충실히 포함하고 있다[10]. 하지만 이러한 Wang 등의 기법도 정적 데이터를 기준으로 설계되었다. 최근 사용되는 클라우드 스토리지는 SaaS 등을 통해 데이터를 수정할 수 있다. 즉 최근의 스토리지는 데이터가 동적으로 변경된다. 공개검증 기법도 이러한 흐름에 맞추어 동

적 업데이트를 제공할 수 있어야 한다. 이때 고려해야 할 사항은 검증자가 유지하고 있는 정보와 업데이트된 데이터의 불일치다. Wang 등은 머클 해쉬 트리 기반 기법을 통해 동적으로 업데이트 할 수 있는 기법을 제안했다. 하지만 이 기법은 보안 측면에서 일부 허점을 포함하고 있어 보완이 필요하다. 향후 연구는 동적 업데이트에 초점을 맞춘 공개검증 기법과 연산량 개선이 필요하다.

참고문헌

[1] M. Lillibridge, S. Elnikety, A. Birrell, M. Burrows, and M. Isard, “A cooperative internet backup scheme,” In Proceedings of the Annual Conference on USENIX Annual Technical Conference, pp. 3-3, 2003.

[2] M. Naor and G.N. Rothblum, “The complexity of online memory checking,” In proceesings of the 46th Annual IEEE Symposium on Foundations of Computer Science (FOCS ’05), pp. 573-584, 2005.

[3] A. Jules and B.S. Jr. Kaliski, “Pors: proofs of retrievability for large files,” In Proceedings of the 14th ACM Conference on Computer and Communications Security (CCS ’07), pp. 584-597, 2007.

[4] F. Sebe, J. Domingo-Ferrer, A. Martinez-Balleste, Y. Deswarte, and J. Quisquater, “Efficient remote data possession checking in critical information infrastructures,” IEEE Transactions on Knowledge and Data Engineering, Vol. 20, No. 8, pp. 1034-1038, 2008.

[5] Y. Deswarte, J. Quisquater, and A. Saidane, “Remote integrity checking,” The Sixth Working Conference on Integrity and Internal Control in Information Systems (IICIS), 2004.

[6] D.G. Filho and P. Barreto, “Demonstrating data possession and uncheatable data transfer,” IACR Cryptology ePrint Archive, 150, 2006.

[7] G. Yamamoto, S. Oda, and K. Aoki, “Fast integrity for large data,” In Proceedings of the ECRYPT, 2007.

[8] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song, “Provable data possession at untrusted stores,” In Proceedings of the 14th ACM Conference on Computer and Communications Security (CCS ’07), pp. 598-609, 2007.

[9] Q. Wang, C. Wang, J. Li, K. Ren, and W. Lou, “Enabling Public Verifiability and Data Dynamics for Storage Security in Cloud computing,” In Proceedings of the 14th European conference on Research in computer security (ESORICS'09), pp. 355-370, 2009.

[10] C. Wang, S.M. Chow, Q. Wang, K. Ren, and W. Lou, “Privacy-Preserving Public Auditing for Secure Cloud Storage,” IEEE Transactions on Computers, Vol. 62, No. 2, pp. 362-375, Feb. 2013.