클라우드 스토리지의 효율적인 이용과 자료에 대한 자기소유권 강화방안

장준언*, 정준권**, 정태명***
*성균관대학교 컴퓨터공학과
**성균관대학교 전자전기컴퓨터공학과
***성균관대학교 정보통신대학
e-mail:jangjoonun@gmail.com, jkjung@imtl.skku.ac.kr, tmchung@ece.skku.ac.kr

Efficient Use of Cloud Storage and the Proposal for Enhancing Possession to Own File

Joon-Un Jang*, Jun-Kwon Jung**, Tai-Myoung Chung***
*Department of Computer Engineering, Sungkyunkwan University
**Department of Electrical and Computer Engineering, Sungkyunkwan University
***College of Information and Communication Engineering, Sungkyunkwan University

요 약

클라우드 서비스는 향후 IT 분야에 있어 중요한 기술이 되고 있다. 특히 모바일 네트워크 시장의 규모가 커감에 따라서 클라우드 서비스는 지금보다 더 부각되는 기술이 될 것이다. 하지만, 사용자의 정보가 제 3자에게 위탁된다는 점에서 보안 문제가 가장 큰 이슈가 되고 있다. 본 논문에서는 자료에 대한 암호화를 통해 개인의 정보소유권을 강화하면서 클라우드 스토리지를 효과적으로 이용할 수 있는 시스템을 제안한다.

1. 서론

클라우드 컴퓨팅의 등장은 컴퓨팅 자원의 활용에 있어 효율적인 자원의 이용이 가능하게 되었고, PC 뿐만 아니라 스마트 디바이스 이용이 급증함에 따라 모바일 클라우드 서비스에도 관심이 집중되고 있다. 클라우드 서비스는 인터넷을 통해 IT 자원을 필요한 만큼 빌려서 사용할 수 있고, 실시간 확장성을 지원받으면서 사용한 만큼의 비용을 지불하는 특징을 갖고 있다. 해외에서는 아마존이나 구글, 마이크로소프트 등의 기업이 주로 클라우드 서비스를 제공하고 있고, 국내에서는 네이버나 다음 등의 포털 사이트가 주로 제공하고 있다.

이러한 클라우드 서비스 중에서도 사용자가 일반적으로 접할 수 있는 서비스는 IaaS(Infrastructure as a Service)이다. 이는 단순히 사용자가 자신의 데이터를 다른 스토리지에 위탁하여 저장하는 서비스를 의미하는 것으로, 이러한 클라우드 서비스의 특성상 데이터가 위탁되어 있다는 점에서 자신의 데이터의 훼손과 기밀유지 등의 보안문제가 발생한다.

이러한 보안문제에 대비하여 데이터 무결성 및 기밀성 유지를 위해 사용자가 자신만의 키(key)를 이용해 데이터를 암호화하여 저장하는 방식이 존재한다. 하지만 이 방식을 이용할 경우 키 교환과 키 관리 문제를 위한 추가적인 정책이 필요하게 될 뿐만 아니라 키가 변경되었을 때의 전체 파일에 대해 다시 암호화를 해야 하는 문제가 발생하게 된다. 또한, 이 방식은 같은 파일을 가지고 있는 유저들이 서로 다른 키를 가지고 암호화를 하기 때문에 서로 다른 파일의 형태로 저장이 되므로, 스토리지 공간에 대한 심각한 낭비를 초래할 수 있다.

본 논문에서는 위와 같은 단점을 개선하기 위해, 다음과 같은 목표를 달성하는 시스템을 제안한다.

1. 암호화, 복호화 과정을 로컬에서만 수행하여, 사용자가 신뢰하는 제 3자와 클라우드 시스템은 결코 사용자의 데이터를 확인할 수 없다.
2. 중복된 파일이 서로 다른 암호방식으로 암호화되어 클라우드 스토리지의 효율을 떨어트리지 않게 한다.
3. 사용자가 관리해야 할 키의 부담을 줄인다.

이어지는 2장에서는 제시된 요구사항을 해결하기 위해 기존에 연구된 논문들과 관련 기술에 대해 기술하며, 3장에서는 실제로 시스템이 어떻게 작동하는지를 설명한다. 마지막 4장에서는 본 시스템이 가지고 있는 오버헤드와 그 해결방안 그리고 향후 연구방향에 대해 논한다.

2. 관련 연구

2.1 파일의 효율적인 저장 방식

George Davida 등이 제안한 기존 연구 방식은 클라우드 스토리지 내의 파일 중복 문제와 효율성에 대한 해결책을

제시하였다[1]. 이 논문에서는 현재 DropBox로 대표되는 클라우드 스토리지에 대한 암호화 방식이 가지고 있는 문제점을 극복하기 위한 방안을 제시하였다. 기존의 DropBox는 사용자의 스토리지 비밀번호를 키로 하여 파일을 암호화하고 저장하는 방식이기 때문에 하나의 키에 대한 의존성이 크다는 점과 키로 사용되는 비밀번호가 회사 내부의 관리자에 의해 유출이 될 수 있는 문제가 발생할 수 있는 문제점이 있다.

따라서 본 논문에서는 사용자의 비밀번호가 아닌 파일 자체를 이용하여 키를 발생시키는 방식을 이용하였고, 이는 유사 파일 저장 방식에 대한 제안을 지원하여 스토리지 공간을 효율적으로 이용하는 방안을 제시한다. 파일 자체를 통한 키를 발생시켜 암호화하게 되면, 파일을 보유하고 있는 사용자들끼리 같은 키를 공유하게 됨으로써, 파일에 대한 유사성을 검증할 수 있게 된다. 이 파일의 유사함의 정도를 위해 Hamming Distance를 이용한 Error Correction방식을 제안한다.

본 논문의 제안 방식을 이용할 경우, 파일 자체를 통해 발생시킨 키를 사용하기 때문에 하나의 키에 대한 의존성을 약화시킬 수 있고, 같은 파일을 가진 사용자끼리 같은 키를 이용하여 암호화를 수행하므로 파일의 유사성 검증이 가능해지고 이를 통해 스토리지 내의 같은 파일의 중복저장에 따른 낭비를 막을 수 있다. 하지만, 저장하게 되는 자료의 양이 많으면 많아질수록 사용자가 관리해야 하는 키의 개수가 많아지며, 키를 파일 형태로 보관하는 경우에는 사용자의 컴퓨터에 문제가 생겨 키를 보관한 파일의 분실의 경우에 대한 리스크가 커지게 된다.

2.2 검색 가능 암호 시스템

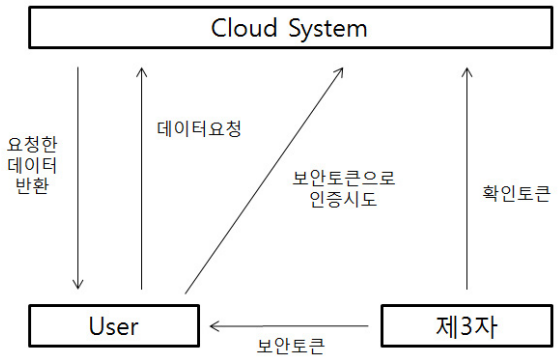
본 논문에서 제안되는 시스템은 사용자에 의해 모든 파일이 암호화되어 저장되기 때문에, 암호화된 파일을 복호화 하지 않고서는, 파일의 검색이 불가능하다. 이는 심각한 비효율적인 문제를 초래하게 된다. 검색 가능 암호 시스템을 이용하게 되면 이 문제를 해결할 수 있다.

검색 가능 암호 시스템이란 사용자가 암호화한 자료에서 복호화를 하지 않고 검색을 하기 위해 사용하는 것이다. 자료를 검색하기 위해서는 키워드를 서버에 제공해야 하며, 일반적으로 자료는 그 자료에 포함된 키워드들의 집합으로 정의된다. 검색 가능 암호 시스템은 키 생성, 암호화, 트랩도어 생성, 검색의 4가지 단계로 이루어지며, 사용자에 의해 생성된 주어진 키워드에 대한 트랩도어를 통해 자료를 검색하는 것을 의미한다. 이를 통해, 완전히 암호화되어 저장된다고 하더라도, 사용자는 전체 파일에 대한 복호화 없이 트랩도어의 수단으로 사용 될 Index File에 대한 복호화만을 통해 자신의 파일을 검색, 확인할 수 있다[2].

위의 두 개의 관련 연구를 통해 본 논문에서 제안할 시스템은 파일을 암호화할 때에 파일블록을 이용해서 발생시킨 키를 바탕으로 암호화하여 파일의 유사성을 검증할 수 있도록 하며, 검색 가능 암호 시스템을 통해 암호화된 파일에 대해서도 미리 등록한 keyword를 통해 검색할 수 있고, 또한 키 관리의 문제를 해결하기 위하여 각각의 파일블록을 암호화하기 위하여 사용된 키들을 자신의 Public key로 암호화하여 index file내부에 저장시키는 방법으로 사용자의 키 관리 부담을 더는 시스템을 제안한다.

3. 제안 모델

제안 모델의 전체적인 시스템은 (그림1)과 같다.



(그림 1) 시스템 개략도

클라우드 스토리지가 암·복호화 과정에 참여하게 되면, 클라우드 내부에서의 Man-In-The-Middle-Attack 에 의해 보안성에 큰 타격을 입을 수 있고, 자료의 유출여부를 사용자가 알기가 어렵기 때문에 그 과정에 참여하지 않는다. 따라서 클라우드 스토리지는 단순히 파일을 저장하는 작업을 수행하며, 분산 파일 시스템으로서의 역할만을 수행하게 된다.

사용자에 의해 신뢰받는 제3자는 올바른 사용자의 인증을 확인하기 위하여 사용하며, 클라우드 스토리지에게 사용자의 Public key를 전송해주는 등의 인증 및 검증 업무를 수행하게 된다.

유저는 제3자에게 인증을 시도하고, 클라우드 스토리지에게 서비스를 요청한다. 또한, 자신이 보관할 자료에 대한 암호화와 보관된 자료에 대한 열람을 위한 복호화를 수행한다.

본 논문에서 제안할 시스템을 위해서는 사용자가 신뢰할 수 있는 제 3자가 있으며, 제 3자와 사용자가 사용하는 통신 채널은 안전하다는 가정이 필요하다. 또한, 설명을 위해 사용되는 용어들은 다음과 같다.

- Enc(Data,Key) : Data를 Key로 암호화
- Dnc(Data,Key) : Data를 Key로 복호화
- Comp(Data) : Data를 압축
- DeComp(Data) : 압축된 Data의 압축해제

3.1 인증

사용자가 클라우드 시스템에 접근할 때 제3자에게 인증을 요청하게 되는데, 제3자는 사용자에게 보안토큰을 사용자의 공개키로 암호화하여 전송하고, 클라우드 시스템에 확인토큰을 전송한다. 사용자는 자신의 비밀키로 제3자로부터 받은 암호화된 토큰을 복호화하고 이를 클라우드 시스템에 전송하여 인증을 시도한다. 시스템은 이 토큰의 유효성을 검증하기 위하여 HMAC-SHA를 이용한다[3].

3.2 파일의 저장

우선 파일을 고정된 크기의 여러 블락으로 나눈다. ($F = m_1, m_2, \cdots, m_n$) 나누어진 파일들의 각각의 블락에 대해 해쉬값, $H(m_1)$, $H(m_2)$, …, $H(m_n)$을 계산한다. 이 해쉬값들은 파일블록의 키로 사용된다. 그 후, $H(H(m_1))$, $H(H(m_2))$, …, $H(H(m_n))$ 해쉬를 두 번 적용한 값들을 계산한다. 이 값들은 파일의 유사성 검사를 위해 사용된다. 이 두 종류의 해쉬값들은 검색 가능 암호 시스템에서 이용할 Index File속에 같이 저장되어 암호화된다. 두 번 해쉬함수를 적용한 값들은 시스템 내부에서 유사성 검증을 이용하여 사용해야 하는 것이므로 암호화를 하지 않고 키로 사용되는 한 번 해쉬함수를 적용한 값들만 사용자의 Public key로 각각의 파일블록에 대해 서로 다른 키를 사용하게 되는 만큼 발생할 수 있는 키의 부담을 인덱스 파일 속에 키를 따로 저장시킴으로써, 사용자의 키 관리 부담을 덜어줄 수 있다. 그리고 각각의 파일블록들은 해당되는 $H(m_n)$으로 암호화한다. 사용자는 최종적으로 인덱스파일과 암호화된 파일블록들을 클라우드 서버에게 저장 요청한다. 저장을 위한 과정은 다음과 같다.

IF = (Enc(H(m_1),Public Key), …, Enc(H(m_n),Public Key), H(H(m_1)), … (H(H(m_n)))
※ IF : Index File
F' = Enc(m_1,H(m_1)), …, Enc(m_n,H(m_n))
Send(IF,F',Operation(Store))

사용자로부터 저장명령을 요청받은 클라우드 시스템은 네임노드에 인덱스 파일을 저장하고 파일블록은 데이터노드에 저장한다.

클라우드 시스템은 저장이 완료된 파일블록의 중복을 검토하기 위해서 총 3개로 중복시켜 저장하게 될 파일블록 중 하나를 바탕으로 파일의 중복성을 검사한다. 중복 검사 후 같은 파일이 존재하면, 네임노드의 메타데이터의 링크를 존재하는 파일로 연결하고, 존재하는 파일을 공유하고 있는 사용자들의 메타데이터의 중복여부를 설정한다. 여기서 중복여부를 설정하는 이유는 해당 데이터를 삭제하는 경우에 공유하고 있는 사용자가 존재한다면 삭제할 수 없기 때문이다. 이를 확인하기 위한 메타데이터의 기본적인 포맷은 다음과 같다.

Metadata (File Name, File Paths(3), Share, Modified)

여기서 Share Field가 공유된 사용자가 존재하는지를 확인하기 위한 field이며, modified는 후에 파일의 수정에서 다뤄질 것이다.

3.3 파일의 삭제

데이터 삭제시에는 복수의 사용자가 동일한 파일을 공유하고 있는지를 확인하여, 동일한 파일 공유자가 존재하는 경우에는, 삭제 명령을 요청한 사용자의 메타데이터만 삭제하고, 파일 공유자가 존재하지 않는 경우에는 실제 파일까지 모두 삭제한다.

3.4 파일의 수정

사용자는 파일을 수정하여 전송할 때, 수정된 파일이 아닌 원본과의 차이를 전송한다. 수정되기 전과 수정한 후의 파일을 Exclusive-OR 하여, 같은 bit는 0, 다른 bit는 1로 된 수정본을 생성한다. 원본과 같은 부분이 많기 때문에 수정본은 대부분의 bit가 0으로 설정되어 있을 것이므로, 효과적으로 압축할 수 있다. 이는 기존에 발표한 논문에서 제안한 Error Correction을 이용한 방식과 유사한 방법이다[1]. 이 수정파일은 사용자의 Public key로 직접 암호화한다. 수정과정은 다음과 같다.

F_{update} = Original F ⊕ Modified F
Comp(F_{update})
Enc(Comp(F_{update}),Public Key)
Send(Enc(Comp(F_{update},Private Key)),Operaton(Modify))

수정요청을 받은 클라우드 시스템은 메타데이터에 수정파일에 대한 정보를 추가한 후, 데이터노드에 수정파일을 저장한다. 해당 사용자는 수정된 파일을 불러오는 경우, 원본파일과, 수정파일을 동시에 불러들여 자신이 수정한 내용을 복원하여 사용하게 된다. 파일을 불러오는 과정은 다음과 같다.

H(m_n) = Dnc(Enc(IF,Private Key), Private Key)
F_1 = Dnc(Enc(F,H(m_n)),H(m_n))
F_2 = Decomp(Dnc(Enc(Comp(F_{update}),Public Key),Private Key))
$F = F_1 \oplus F_2$

3.5 보안성 평가

파일의 암·복호화 과정이 오직 로컬에서만 진행되기 때문에 클라우드 시스템이 중간에 파일을 훔쳐볼 수 없다. 또한 이 과정에서 사용되는 키 역시도 사용자가 생성한 자신의 공개키로 암호화하여 저장하기 때문에, 사용자를 제외하고는 복호화를 수행할 수 없다. 이는 클라우드 시스

템 내에 저장되는 파일의 기밀성을 증대시키는 효과를 가져 온다. 단, 파일블록마다 서로 다른 키를 사용하기 때문에, 사용자가 관리해야 할 키의 부담이 많아져 분실이나 노출의 우려가 있지만 이는 파일을 암호화하는데 직접적으로 사용하는 키를 자신의 개인키로 암호화하여 클라우드 스토리지에 직접 저장하는 것으로 해결할 수 있다.

3.6 성능평가

파일의 암·복호화가 오로지 로컬에서만 수행되므로, 로컬에 입장에서는 기존의 클라우드 시스템보다 다소 성능이 떨어질 수 있다. 하지만 전체적으로 기존의 클라우드 시스템이 중앙 집중식 암호화 처리 방식보다 더 빠르며 더 강한 보안을 제공할 수 있다. 뿐만 아니라 같은 파일에 대해서 서로 같은 키로 암호화하기 때문에, 한 파일을 보유자에 한해서 공유시킬 수 있고, 이는 클라우드 스토리지의 효율적인 운영에 도움을 줄 수 있다.

4. 결론

4.1 요약

제안된 논문에서는 파일의 암·복호화를 반드시 로컬에서만 수행할 수 있게 하여 클라우드 시스템이 파일을 확인할 수 있는 방법을 봉쇄하였다. 또한, 파일블록으로 발생시킨 키를 바탕으로 암호화하므로 같은 파일을 가지고 있는 사람들은 모두 같은 키를 바탕으로 암호화하게 되고 따라서 클라우드에서는 같은 파일을 가진 사람들끼리 공유가 가능하게 되어 용량을 효율적으로 사용할 수 있게 하였다. 이 때, 파일블록마다 서로 다른 키를 사용하게 됨으로서 생기는 키 관리의 부담을 덜기 위해 인덱스에 키를 같이 저장하고, 이 키를 전체적으로 관리할 수 있는 하나의 키만을 관리하게 하였다.

4.2 오버헤드

파일의 암·복호화가 로컬에서만 진행되기 때문에, 이 과정은 전적으로 사용자의 부담이 된다. 하지만, 정보의 위탁이라는 클라우드 시스템이 가지고 있는 태생적인 보안 문제를 해결하기 위해서는, 이 과정이 반드시 필요하다. 따라서 이 과정의 오버헤드를 최소화하기 위한 효율적인 암호 알고리즘의 선택이 필요할 것이다.

또한, 파일의 유사성 검증을 위해 파일블록마다 서로 다른 키를 사용하기 때문에, 각각의 키를 클라우드 스토리지가 감당해야 하는 공간적 오버헤드가 발생한다. 하지만, 이것이 같은 파일을 중복 저장시키지 않아서 스토리지 내의 공간 효율을 높일 수 있는 목적이 있는 만큼, 이 오버헤드는 무시할 수 있을 것이다.

마지막으로, 파일에 대한 수정본을 따로 저장하게 되는 오버헤드는 수정본은 원본 파일과 많은 차이가 발생하지 않기 때문에, 대부분이 0비트로 설정되어 효과적으로 압축할 수 있어 사용되는 공간을 줄일 수 있다. 또한, 수정본이 원본과 일정량 이상의 차이가 발생하게 되면, 새로운 파일로 인식하게 되는 방식을 이용하여 문제를 해결할 수 있다.

4.3. 향후 연구

본 논문에서 제안된 시스템이 발생시키는 오버헤드의 최소화 방안에 대한 연구가 필요하다. 가장 큰 오버헤드가 발생하는 부분으로 예상되는 클라우드 시스템의 유사성 검사를 빠르게 수행할 수 있는 기술이 선행되어야 할 것이다.

참고문헌

[1] George Davida and Yair Frankel “Efficient Encryption and Storage of Close Distance Messages with Applications to Cloud Storage”, Springer-Verlag Berlin Heidelberg, 2012, 465-473

[2] 조남수, 홍도원 “검색 가능 암호 시스템 기술 동향”, ETRI, 2008

[3] OSPFv2 HMAC-SHA Cryptographic Authentication, RFC5709, 2009

[4] Hsiao-Ying Lin, Shiuan-Tzuo Shen, Wen-Guey Tzeng, Bao-Shuh P.Lin “Toward Data Confidentiality via Integrating Hybrid Encryption Schemes and Hadoop Distributed File System”, 26th IEEE International Conference on Advanced Inforamtion Networking and Applications, 2012, 740-747