프록시 재암호화 기반의 안전한 클라우드 스토리지 데이터 관리 기법

고웅*, 곽진**
*순천향대학교 정보보호학과 정보보호응용및보증연구실
**순천향대학교 정보보호학과
e-mail:wgo@sch.ac.kr, jkwak@sch.ac.kr

A Cloud Storage Data Management based on Proxy Re-encryption

Woong Go*, Jin Kwak**
*ISAA Lab, Dept of Information Security Engineering, Soonchunhyang University
**Dept of Information Security Engineering, Soonchunhyang University

요 약

클라우드 스토리지는 사용자의 요구 또는 데이터의 이용 추이에 따라 자원의 분배를 탄력적으로 제공함으로써 컴퓨팅 자원에 대한 효율적으로 관리가 가능하다. 이에 최근 다양한 서비스가 클라우드 스토리지를 제공하고 있으나, 개인적인 데이터를 소유자가 직접 관리하기 어려워 데이터의 보안성을 보장할 수 없다. 따라서 본 논문에서는 클라우드 스토리지에 저장되는 데이터에 대하여 소유자만이 복호화가 가능하도록 프록시 재암호화를 이용하여 키를 분배하고 데이터를 관리하는 기법을 제안한다. 이를 통해 데이터의 보안성 및 제 3자에 의한 유출을 차단할 수 있다.

1. 서론

클라우드 스토리지는 네트워크를 이용하여 특정 공간에 설치된 가상화된 저장 공간을 사용하며, 사용자가 특정 저장 디바이스를 구매하여 소유할 필요없이 사용한 만큼만 비용을 지불하여 효율성을 향상시킨 서비스이다[1]. 이를 통해 사용자는 자신의 자료를 언제 어디서든지 클라우드 스토리지에 저장하고 불러와 사용할 수 있다. 그러나 외부에 데이터가 존재하므로 데이터 유출이나, 접근제어 등의 직접적인 데이터 통제가 어렵다. 이는 데이터에 대한 보안성을 완벽하게 보장할 수 없다는 것을 의미한다[2].

따라서 본 논문에서는 클라우드 스토리지에 저장되는 데이터에 대한 접근 시 정당한 사용자만이 데이터를 복호화하여 확인할 수 있도록 프록시 재암호화를 이용한 암호화키 분배와 안전한 데이터 관리 기법을 제안한다. 프록시 재암호화는 송신자의 개인키와 수신자의 공개키를 이용하여 재암호화키를 생성한 후, 송신자의 공개키로 암호화된 데이터를 복호화 없이 수신자의 개인키로 복호화할 수 있도록 하는 알고리즘이다[3].

본 논문의 구성은 다음과 같다. 2장에서는 클라우드 스토리지의 문제점을 분석하고, 3장에서는 안전한 데이터 관리 기법을 제안한다, 4장에서 보안성 및 효율성을 분석하고, 마지막으로 5장을 결론으로 끝을 맺는다.

이 논문은 2012년도 정부(교육과학기술부)의 재원으로 한국연구재단의 기초연구사업 지원을 받아 수행된 것임. (2012-010886)

2. 클라우드 스토리지 문제점 분석

2.1 데이터 통제 불가

사용자는 클라우드 스토리지에 저장된 자신의 데이터가 어디에 저장되고 어떻게 관리되는지 정확히 알지 못한다. 또한, 데이터의 저장이 모두 클라우드 스토리지에서 이루어지므로 사용자가 자신의 데이터에 대한 직접적인 통제가 불가능하다. 이는 클라우드 스토리지에 저장되는 데이터가 사용자가 동의 없이 임의로 제3자에게 제공되거나, 내부자에 의해 불법적으로 접근되더라도 사용자가 이를 알아 낼 수 없다는 것을 의미한다.

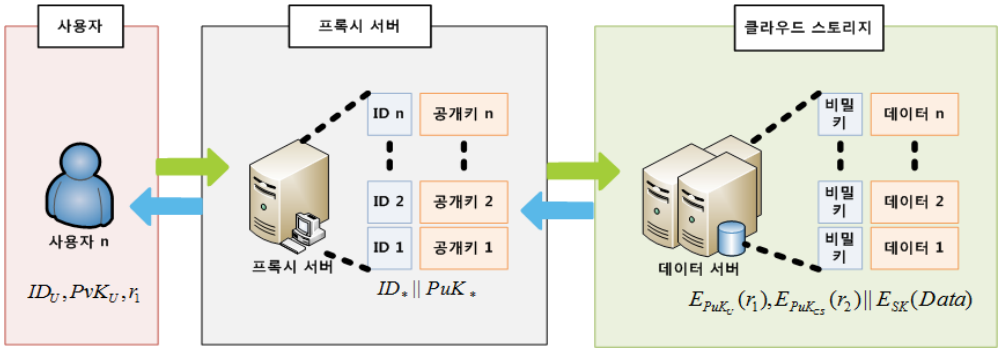
2.2 데이터 암호화키 문제

클라우드 스토리지는 다수의 사용자가 함께 사용하므로 상당히 많은 데이터가 저장된다. 이러한 데이터의 보안성을 보장하기 위해서 모두 암호화하여 저장되지만, 이를 위한 키를 모두 개별적으로 관리하는 것은 시간적, 비용적 측면에서 비합리적이다. 따라서 각 사용자별로 상이한 키를 이용하여 소유자의 데이터를 암호화하는 실정이다. 그러나 이렇게 하나의 키로 다수의 데이터를 암호화할 경우, 해당 키가 노출되면 특정 사용자의 모든 데이터를 알아낼 수 있는 문제가 발생한다.

3. 제안 기법

3.1 기본 구조

본 논문에서는 클라우드 스토리지의 데이터를 안전하게 관리하기 위하여 프록시 재암호화를 이용하여 암호화



(그림 1) 제안 기법 구조도

키를 분배하고 정당한 사용자만 데이터를 확인할 수 있도록 하는 데이터 관리 기법을 제안한다. 본 기법은 데이터 저장 단계와 데이터 요청 단계로 구성된다.

3.2 용어 정의

다음 표에 제안하는 기법에서 사용되는 용어들의 표기법 및 정의를 기술한 것이다.

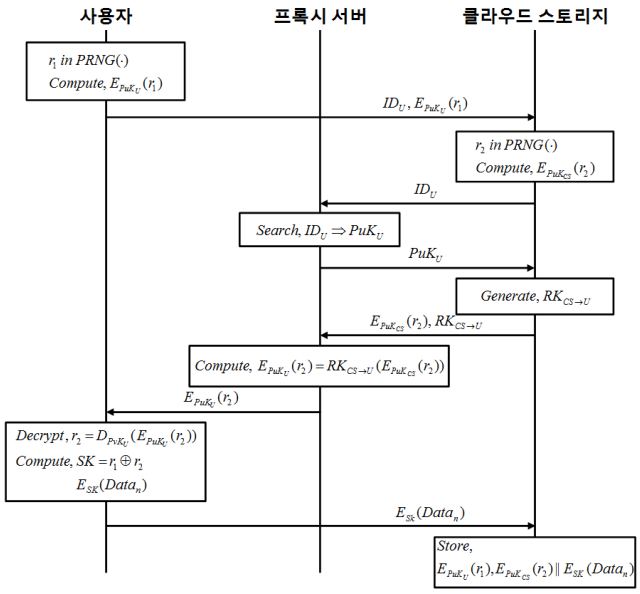
<표 1> 용어 정의

표기법	정의		
U	사용자	FS	프록시 서버
ID_U	사용자 ID	CS	클라우드 스토리지
r_n	난수값 (n=1,2)	$PRNG(\bullet)$	의사난수생성기
$Data_n$	저장된 사용자 데이터		
PuK_U, PvK_U	사용자 공개키, 개인키(서명)		
PuK_{CS}, PvK_{CS}	클라우드 스토리지 공개키, 개인키(서명)		
SK	데이터 암복호화키		
$RK_{CS \to U}$	프록시 재암호화키		

3.3 데이터 저장 단계

데이터 저장 단계에서는 사용자의 데이터($Data_n$)를 암복호화하기 위한 키(SK) 생성을 위해 사용자(U)와 클라우드 스토리지(CS)가 난수(r_1, r_2)를 생성하여 전송하며, 안전한 전송을 위해 프록시 재암호화를 거친다. 마지막으로 U가 데이터를 암호화하여 CS에게 전송하여 저장한다. 그림 2는 데이터 저장 단계를 나타내는 프로토콜이다.

각 단계에 대해 설명하면, 사용자는 안전하게 자신의 데이터를 전송하고 저장하기 위해 r_1을 생성하여 SK를 생성한다. 그 후, 자신의 공개키(PuK_U)를 이용하여 이를 암호화하고 사용자 ID(ID_U)와 함께 CS에게 전송한다. 사용자가 생성한 r_1은 사용자의 개인키(PvK_U) 없이 누구도 알아 낼 수 없으며, 암복호화키는 매번 새롭게 생성되므로 키 노출이 발생하더라도 사용자의 다른 데이터의 보안성을 보장할 수 있다.



(그림 2) 데이터 저장 단계 프로토콜

사용자(U) -> 클라우드 스토리지(CS)

$r_1 \in PRNG(\bullet)$
$Compute, E_{PuK_U}(r_1)$
$Send\ to\ U: ID_U, E_{PuK_U}(r_1)$

CS는 데이터를 전송받은 후 마찬가지로 SK를 생성하기 위해 r_2를 생성하고, 이를 CS의 공개키(PuK_{CS})로 암호화한다. 그리고 사용자에게 받은 ID_U를 프록시 서버(FS)에 전송하여 사용자의 공개키를 요청한다. 프록시 서버는 해당 ID_U를 통해 사용자의 공개키를 알려준다. 사용자의 ID_U와 공개키는 노출되어도 문제가 없으며, 임의로 변경하더라도 최초 사용자가 생성한 r_1을 복호화하기 위한 개인키를 알 수 없으므로 SK를 생성할 수 없다.

클라우드 스토리지(CS) -> 사용자(U)

$r_2 \in PRNG(\bullet)$
$Compute, E_{PuK_{CS}}(r_2)$
$Send\ to\ U: ID_U$

프록시 서버(FS) -> 클라우드 스토리지(CS)

$Search, ID_U \Rightarrow PuK_U$
$Send\ to\ CS: PuK_U$

CS는 전송받은 사용자의 공개키와 자신의 개인키를 이용하여 재암호화키($RK_{CS \to U}$)를 생성한다. 해당 키는 FS에서 CS가 생성한 난수값을 U가 복호화할 수 있도록 재암호화하는데 이용된다. 재암호화키 역시 공격자가 CS의 개인키를 알아야하므로 제3자가 만들어낼 수 없다.

클라우드 스토리지(CS) -> 프록시 서버(FS)

$Generate, RK_{CS \to U}(PuK_U, PvK_{CS})$
$Send\ to\ FS: E_{PuK_{CS}}(r_2), RK_{CS \to U}$

FS는 CS가 공개키로 암호화하여 전송한 암호화된 r_2를 재암호화키를 이용하여 U의 개인키로 복호화할 수 있도록 변환을 수행한다. 이 후, 생성된 정보를 사용자에게 전송한다.

프록시 서버(FS) -> 사용자(U)

$Compute, E_{PuK_U}(r_2) = RK_{CS \to U}(E_{PuK_{CS}}(r_2))$
$Send\ to\ U: E_{PuK_U}(r_2)$

사용자는 암호화된 r_2을 복호화하여 알아낸 후, 자신이 생성한 r_1과 Exclusive-Or 연산하여 SK를 생성한다. 이 키는 전송되는 데이터에 따라 다르므로 데이터별 보안성을 보장한다. 사용자는 암호화된 데이터를 클라우드 스토리지에 전송하고, 클라우드 스토리지는 암호화된 $Data_n$와 두 개의 난수값을 연접하여 저장한다. 이와 같이 저장하면 추후 데이터 복원을 위한 키를 따로 관리할 필요가 없다.

사용자(U) -> 클라우드 스토리지(CS)

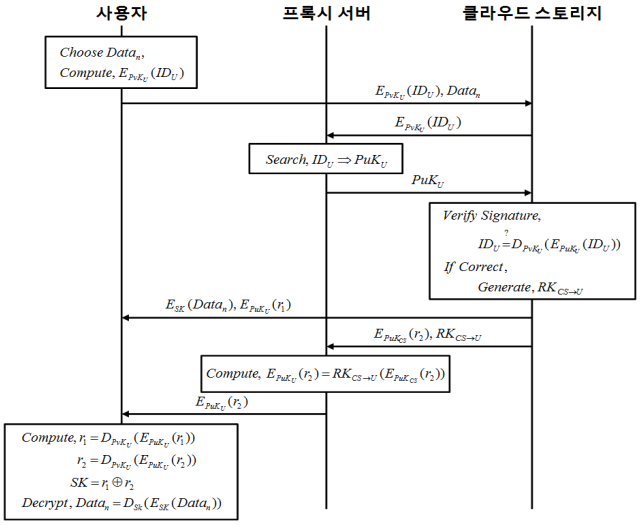
$Decrypt, r_2 = D_{PvK_U}(E_{PuK_U}(r_2))$
$Compute, SK = r_1 \oplus r_2, E_{SK}(Data_n)$
$Send\ to\ CS: E_{SK}(Data_n)$

클라우드 스토리지(CS)

$Store,$
$E_{PuK_U}(r_1), E_{PuK_{CS}}(r_2) \| E_{SK}(Data_n)$

3.4 데이터 요청 단계

데이터 요청 단계에서는 U가 자신이 필요한 데이터를 CS에게 요청하는 단계로, 정당한 사용자의 요청임을 확인시키기 위하여 자신의 ID를 개인키로 서명하여 전송한다. CS는 FS로부터 받은 사용자의 공개키로 서명을 검증하고, 정당한 사용자임을 확인하면 데이터를 전송하여, U가 SK를 생성하여 데이터를 확인할 수 있도록 한다. 그림 3은 데이터 요청 단계를 나타내는 프로토콜이다.



(그림 3) 데이터 요청 단계 프로토콜

각 단계에 대해 설명하면, U는 자신이 필요한 데이터를 요청하기 위해 특정 데이터를 선택하고 이를 CS에게 ID_U와 함께 요청한다. 이 때, 이를 요청한 사용자가 정당한 사용자임을 확인하기 위하여 자신의 개인키를 이용하여 ID에 서명하여 전송한다.

사용자(U) -> 클라우드 스토리지(CS)

$Choose\ Data_n,$
$Compute, E_{PvK_U}(ID_U)$
$Send\ to\ CS: E_{PvK_U}(ID_U), Data_n$

CS는 서명된 사용자의 ID를 FS에 전송하여 공개키를 요청하고, 프록시 서버는 이를 검색하여 CS에 전송한다.

클라우드 스토리지(CS) -> 프록시 서버(FS)

$E_{PvK_U}(ID_U)$

프록시 서버(FS) -> 클라우드 스토리지(CS)

$Search, ID_U \Rightarrow PuK_U$
$Send\ to\ CS: PuK_U$

CS는 전송받은 공개키를 통해 U의 서명값이 올바른지 검증하고, 정당한 사용자임을 확인하면 SK 생성을 위한 r_2를 U가 복호화할 수 있도록 재암호화키를 생성한다. 이 후, CS는 U에게 $Data_n$와 사용자가 생성한 r_1을 전송하고, FS에게는 CS가 생성한 r_2와 $RK_{CS \to U}$를 전송한다.

클라우드 스토리지(CS)

$Verify\ Signature,$
$ID_U \stackrel{?}{=} D_{PuK_U}(E_{PvK_U}(ID_U))$
$If\ Correct,$
$\geq \neq rate, RK_{CS \to U}$

클라우드 스토리지(CS) -> 사용자(U)

$E_{SK}(Data_n)$, $E_{PuK_U}(r_1)$

클라우드 스토리지(CS) -> 프록시 서버(FS)

$E_{PvK_{CS}}(r_2)$, $RK_{CS \to U}$

FS는 전송받은 $RK_{CS \to U}$를 이용하여 암호화된 r_2를 재암호화하여 U에게 전송한다. FS와 통신상에서 U와 CS가 생성한 난수값은 복호화되지 않고, 유출되더라도 개인키를 알 수 없어 복호화가 불가능하다. 또한, CS 역시 사용자의 난수값을 알아낼 수 없어서 SK를 생성할 수 없다.

프록시 서버(FS) -> 사용자(U)

$Compute$, $E_{PuK_U}(r_2) = RK_{CS \to U}(E_{PuK_{CS}}(r_2))$

$Send\ to\ U$: $E_{PuK_U}(r_2)$

마지막으로 사용자는 전송받은 두 개의 난수값을 복호화하고, Exclusive-OR 연산을 통해 SK를 생성한다. 그리고 최종적으로 선택하였던 $Data_n$을 복호화하여 이용한다.

사용자(U)

$Compute$, $r_1 = D_{PvK_U}(E_{PuK_U}(r_1))$

$r_2 = D_{PvK_U}(E_{PuK_U}(r_2))$

$SK = r_1 \oplus r_2$

$Decrypt$, $Data_n = D_{SK}(E_{SK}(Data_n))$

4. 보안성 및 효율성 분석

4.1 데이터 통제 가능

본 논문에서 제안한 방식은 데이터의 암복호화에 사용자가 직접 개입하므로, 데이터에 대한 통제가 가능하다. 기존의 경우, 데이터의 암복호화가 클라우드 스토리지 내에서 일방적으로 이루어져 사용자는 데이터의 기밀성을 보장하는데 역할이 미비하였지만, 본 제안방식에서는 데이터의 암호화를 위한 암복호화키(SK) 생성에 사용자와 클라우드 스토리지가 난수값(r_1, r_2)을 이용한다. 따라서 클라우드 스토리지의 일방적인 암복호화가 아닌 데이터 통제에 직접적인 영향을 미치는 것이다. 이와 같은 방식을 통해 클라우드 스토리지 뿐만 아니라 제3자는 사용자가 생성한 난수값(r_1)을 알 수 없으므로 임의로 데이터를 암복호화 하지 못한다. 데이터 요청 단계에서는 정당한 사용자만 접근이 가능하도록 하기 위하여 사용자의 개인키(PvK_U)를 통해 사용자 ID(ID_U)를 서명함으로써 이를 검증할 수 있다. 예를 들어 공격자가 전송되는 데이터($E_{PvK_U}(ID_U)$, $Data_n$)를 가로채 요청 데이터 또는 ID를 변경하더라도 정상적인 사용자임을 검증할 수 있다.

4.2 데이터 암호화키 보안

데이터를 암호화하기 위한 키는 블록암호 알고리즘을 이용한 대칭키를 사용하므로 사전에 키를 분배하는 문제가 중요하다. 본 제안방식은 데이터 암복호화키를 사용자와 클라우드 스토리지의 난수값(r_1, r_2)를 이용하여 생성하므로, 한쪽의 정보만으로는 생성이 불가능하다. 만약 공격자가 해당 난수값을 임의로 생성하여 전송하려고 해도 재암호화를 위한 키를 생성할 수 없으므로, 사용자측에서 해당 값을 무시할 수 있다.

또한, 해당 난수값을 전송하는데 보안성을 위하여 프록시 재암호화를 이용함으로써 암호화된 난수값을 복호화하지 않고 안전하게 전송할 수 있다.

5. 결론

현재 다양한 클라우드 스토리지 서비스가 제공되고 있으나, 자신의 데이터에 대한 통제가 불가능함에 따라 데이터 보안성이 문제가 되고 있다. 이를 해결하기 위하여 본 논문에서는 프록시 재암호화 기반의 안전한 클라우드 스토리지 데이터 관리 기법을 제안하였다. 제안 기법은 데이터 전송 단계와 데이터 요청 단계로 구성되어 있으며, 데이터 암호화를 위한 키를 프록시 재암호화를 통해 공유하여 보안성을 보장하고 있다.

참고문헌

[1] Kun Liu, Long-jiang Dong, "Research on Cloud Data Storage Technology and Its Architecture Implementation“, Procedia Engineering, Vol.29, pp.133-137, 2012.

[2] Lluis Pamies-Juarez, Pedro García-López, Marc Sánchez-Artigas, Blas Herrera, "Towards the design of optimal data redundancy schemes for heterogeneous cloud storage infrastructures", Computer Networks, Vol.55, Issue.5, pp.1100-1113, 2011.

[3] Ateniese, G., Fu, K., Green, M., Hohenberger, S., "Improved Proxy Re-encryption Schemes with Applications to Secure Distributed Storage", In Proceedings of the 12th Annual Network and Distributed System Security Symposium, pp.29-44, 2005