Journal of The Korea Institute of Information Security & Cryptology ISSN 1598-3986(Print)
VOL.25, NO.1, Feb. 2015 ISSN 2288-2715(Online)
http://dx.doi.org/10.13089/JKIISC.2015.25.1.83

안전하고 효율적인 클라이언트 사이드 중복 제거 기술*

박 경 수,† 엄 지 은, 박 정 수, 이 동 훈‡
고려대학교 정보보호대학원

Secure and Efficient Client-side Deduplication for Cloud Storage*

Kyungsu Park,† Ji Eun Eom, Jeongsu Park, Dong Hoon Lee‡
Graduate School of Information Security, Korea University

요 약

중복 제거 기술(Deduplication)은 동일한 데이터에 대하여 중복 저장을 방지하는 기법으로 클라이언트(Client)와 클라우드 서버(Cloud Server) 간에 데이터를 저장하고 관리하는데 있어 효율성을 제공한다. 하지만 공개된 환경의 클라우드 서버에 데이터를 저장하고 관리하기 때문에, 클라이언트가 저장한 데이터에 대한 프라이버시 문제가 발생할 뿐만 아니라 데이터의 손실이 발생할 수도 있다. 최근 이러한 문제점들을 해결하기 위해 안전한 중복 제거 기술이 제안되었지만, 여전히 각각의 공격의 안전성에 대한 문제가 발생할 뿐만 아니라 비효율적이다.

본 논문에서는 2013년 Bellare 등이 제안한 기법의 키 서버(Key Server)와 질의-응답 메커니즘(Challenge-Response)을 이용하여 안전하고 효율적인 클라이언트 사이드 중복 제거 기술을 제안한다. 제안 기법은 클라이언트 사이드 중복 제거 기술에서 발생하는 다양한 공격에 대해 안전성을 제공하며, 크기가 큰 데이터를 업로드하는 환경에서 높은 효율성을 제공한다.

ABSTRACT

Deduplication, which is a technique of eliminating redundant data by storing only a single copy of each data, provides clients and a cloud server with efficiency for managing stored data. Since the data is saved in untrusted public cloud server, however, both invasion of data privacy and data loss can be occurred. Over recent years, although many studies have been proposed secure deduplication schemes, there still remains both the security problems causing serious damages and inefficiency.

In this paper, we propose secure and efficient client-side deduplication with Key-server based on Bellare et. al's scheme and challenge-response method. Furthermore, we point out potential risks of client-side deduplication and show that our scheme is secure against various attacks and provides high efficiency for uploading big size of data.

Keywords: Cloud storage, Client-side deduplication, Data privacy, Poison attack

I. 서 론

1.1 개요

빅 데이터(Big data) 시대를 맞아 클라이언트들은 클라우드 스토리지 서비스(Cloud Storage Service)를 이용하여 용량이 크고 방대한 양의 데이터를 효율적으로 관리한다. 중복 제거 기술(Deduplication)은 효율성 측면에서 클라우드 스토리지 환경에 가장 큰 영향을 미치고 있는 기술로 동일

접수일(2014년 10월 15일), 수정일(2014년 12월 22일), 게재확정일(2015년 1월 28일)
* 이 논문은 2012년도 정부(교육과학기술부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(No.2012-08697)
† 주저자, parkhaha2@naver.com
‡ 교신저자, donghlee@korea.ac.kr(Corresponding author)

한 데이터를 하나만 저장하여 중복 저장을 방지하는 기술이다. 데이터와 연관성을 가지는 작은 태그(Tag) 값을 이용하여 클라우드 서버는 원본 데이터 하나만 스토리지에 저장하고, 동일한 데이터에 대해서는 태그 값과 함께 클라이언트의 포인터(Pointer)를 모아 저장한다. 이를 통해 클라우드 서버는 90%이상의 저장 공간을 확보할 수 있으며 더불어 통신 대역폭 효율성을 얻을 수 있기 때문에, 현재 대부분의 클라우드 스토리지 기술에 사용되고 있다[8,12].

데이터 중복 제거 기술은 중복 제거를 어떤 방식으로 적용하느냐에 따라 다음과 같이 두 가지 기법으로 나눌 수 있다[1]. 첫 째, 서버 사이드 중복 제거 기술(Server-side Deduplication)은 동일한 데이터에 대한 중복 제거를 클라우드 서버가 진행하며 클라이언트는 매번 데이터를 업로드 하지만 자신의 데이터가 중복 제거가 일어났는지 일어나지 않았는지의 여부는 알 수 없는 기법이다. 이 기법은 클라우드 서버의 저장 공간 이용을 향상 시킬 수 있지만 통신 대역폭의 효율성은 얻을 수 없다. 둘 째, 클라이언트 사이드 중복 제거 기술(Client-side Deduplication)은 데이터를 전송하기 전에 클라이언트가 작은 태그 값을 생성하여 질의-응답(Challenge-Response) 메커니즘을 통해 데이터가 클라우드 서버에 저장되어 있는지 여부를 먼저 확인하는 기법이다. 만약 클라우드 서버에 데이터가 이미 저장되어 있다면, 데이터 전송은 일어나지 않고 클라우드 서버는 기존에 저장되어 있는 데이터에 클라이언트의 포인터를 저장한다. 따라서 이 기법은 클라우드 서버의 저장 공간 이용뿐만 아니라 통신 대역폭의 효율성도 얻을 수 있는 기법이다.

클라우드 스토리지 서비스를 제공하는 많은 기업(DropBox, MozyHome, Memopal 등)들은 저장 공간의 효율성과 통신 대역폭의 향상을 위한 방법으로 클라이언트 사이드 중복 제거 기술을 적용하고 있다. 그러나 일반적으로 클라이언트들은 데이터를 암호화하지 않는 경우가 대다수이고, 스토리지 서버에서 일괄적인 암호화를 수행하기 때문에 데이터 정보가 노출될 위험이 크다. 뿐만 아니라, 공격자가 태그 값 검색 기능을 이용하여 클라우드 서버에 저장된 데이터 정보를 알게 됨으로써 데이터에 대한 프라이버시 문제가 발생하고, 악의적인 클라이언트가 데이터를 의도적으로 변형하여 저장함으로써 이후 클라이언트는 원본 데이터에 대한 손실이 발생하는 문제점이 제기되고 있다.

1.2 관련 연구

2002년, Douceur 등은 암호화된 데이터에 클라우드 서버가 데이터 중복 제거 기술을 적용할 수 있는 컨버전트 암호(Convergent Encryption)기법을 제안하였다[3]. 이 기법은 데이터를 해쉬해서 얻은 값을 비밀키로 이용하여 데이터를 암호화한다. 동일한 데이터에 대해서는 비밀키가 같게 되고, 이 비밀키와 결정적(Deterministic) 암호 알고리즘을 사용하면 동일한 암호문이 생성된다. 따라서 클라우드 서버는 데이터의 정보는 알 수 없지만 중복 제거 기술을 적용할 수 있다. 하지만 엔트로피가 낮거나 작은 도메인을 가지는 데이터에 대해서, 공격자는 후보가 되는 데이터의 해쉬 값 연산을 통해 비밀키를 얻을 수 있기 때문에 전수 조사 공격이 가능한 문제점이 발생한다.

2010년, Harnik 등은 컨버전트 암호 기법을 이용하여 저장할 데이터를 암호화하고, 시스템에 설정된 임계치만큼 반복하여 데이터를 저장하는 기법을 제안하였다[1]. 하지만 이는 동일한 데이터를 하나만 저장하고자하는 중복 제거 기술의 목적과 맞지 않을 뿐만 아니라, 공격자가 변형된 데이터를 임계치만큼 반복하여 저장한다면 이후 클라이언트가 데이터를 잃어버리는 것을 막을 수 없다.

2011년, Halevi 등은 실제 데이터에 대한 접근 권한이 없는 공격자로부터 클라우드 서버에 저장된 데이터의 정보 유출을 막기 위한 데이터 소유권 증명(Proofs of Ownership)기법을 제안하였다[7]. 이 기법에서는 실제 데이터를 소유한 클라이언트가 데이터로부터 생성된 정보를 이용하여 클라우드 서버에게 받은 챌린지에 대한 응답을 전송하여 소유권을 증명한다. 하지만 클라이언트는 데이터를 암호화하지 않고 인코딩 해서 저장하기 때문에 데이터 정보가 노출되는 프라이버시 문제가 발생할 뿐만 아니라, 데이터 소유권을 증명하기 위해 많은 연산을 필요로 하는 단점이 있다.

2013년, Xu 등은 엔트로피가 낮거나 작은 도메인을 가지는 데이터에 대해서 전수 조사 공격에 안전한 기법을 제안하였다[5]. 이 기법은 데이터를 업로드하는 최초의 클라이언트가 임의의 키를 생성하여 데이터를 암호화하기 때문에 암호문에 대한 전수조사 공격을 효과적으로 막을 수 있다. 하지만, 데이터를 해쉬해서 얻은 값을 태그로 사용하기 때문에 공격자가 특정 데이터의 태그를 생성하여 검색 기능을 이용하면 데이터의 저장 여부를 알 수 있게 되어 데이터에 대한 프라

이버시 문제가 발생한다. 또한 악의적인 클라이언트(최초 업로더)로부터 변형된 데이터에 대한 암호문이 저장되었을 경우, 이를 확인할 수 있는 방법이 없기 때문에 데이터에 대한 손실이 발생한다.

2014년, Shin 등은 키워드 검색을 제공하는 공개키 암호화(Public-key Encryption with Keyword Search) 기법[10]에 기반하여 두 가지의 동등 술어 암호화(Equality Predicate Encryption) 기법을 설계하였고, 이 기법들을 이용하여 클라이언트와 클라우드 서버 간의 질의-응답 메커니즘을 구성함으로써 효율적이고 안전한 중복 제거 기술을 제안하였다[9]. 하지만 이 기법은 데이터를 해쉬한 값을 비밀키로 사용하기 때문에 여전히 전수 조사 공격에 취약하다[11]. 또한, 원본 데이터 손실을 막기 위해 r개의 데이터를 저장하여 관리하는데, 이는 동일한 데이터를 하나만 저장하여 저장 효율성을 제공하는 중복 제거 기술의 목적과 상충하는 결과이다.

한편, 2013년, Bellare 등은 엔트로피가 낮거나 작은 도메인을 가지는 데이터들에 대한 전수 조사 공격을 막기 위해 비밀키 질의 횟수를 제한하는 정책을 가진 키 서버(Key Server)를 두고 RSA 블라인드 서명(RSA Blind Signature) 기법[4]을 이용하여 안전하게 데이터의 비밀키를 생성하는 서버 사이드 중복 제거 기술을 제안하였다[6].

1.3 기여도

클라이언트 사이드 중복 제거 기술은 클라이언트가 데이터를 업로드 하기 전에 태그 값을 이용하여 데이터의 저장 여부를 확인한다. 이때, 공격자는 태그 값을 이용한 검색 기능을 사용하여 특정 데이터의 저장 여부를 알 수 있을 뿐만 아니라, 채널 상의 태그를 가로채 클라우드 서버에 전송함으로써 다운로드 권한을 획득하여 데이터의 정보를 알 수 있는 데이터에 대한 프라이버시 문제가 발생한다. 또한, 태그와 암호문은 독립적으로 생성되기 때문에 동일한 메시지로부터 생성되었는지 확인할 수 있는 방법이 없다면, 공격자(최초 업로더)가 변형된 데이터를 저장했을 때, 이후 클라이언트는 원본 데이터를 손실하는 문제가 발생한다. 이 공격은 포이즌 공격(Poison Attack) 또는 중복 위장 공격(Duplicate Faking Attack)이라고 정의되었으며 이에 대한 안전성을 제공하는 클라이언트 사이드 중복 제거 기술은 없다.

본 논문에서는 위 문제점을 모두 해결할 수 있는 안전하고 효율적인 클라이언트 사이드 중복 제거 기술을 제안한다. 데이터에 대한 프라이버시를 제공하기 위해 Bellare 등이 제안한 개념을 적용하여 키 서버로부터 생성된 비밀 값을 이용하여 데이터에 대한 비밀키와 태그를 생성하도록 구성한다. 키 서버와의 통신으로부터 비밀키를 안전하게 생성하여 데이터에 대한 전수조사 공격을 막을 수 있고, 공격자가 검색 기능을 이용해 특정 데이터의 저장 여부를 알 수 없도록 하며, 안전하게 생성된 비밀키를 이용하여 데이터 암호화를 진행해 데이터에 대한 프라이버시 문제를 해결한다.

또한, 데이터 업로드 단계에서 클라이언트와 클라우드 서버 간의 양방향 질의-응답 메커니즘을 수행하여 클라우드 서버는 해당 암호문이 제대로 저장되어 있음을 증명하고, 클라이언트는 데이터 소유권에 대한 증명을 통해 데이터 다운로드 권한을 얻는다.

본 논문의 구성은 다음과 같다. Ⅱ장에서는 본 논문에서 제안한 기법의 시스템 모델에 대해 설명하고, Ⅲ장에서는 클라이언트 사이드 중복 제거 기술에서 고려해야하는 공격 모델에 대해 설명한다. Ⅳ장에서는 제안 기법 설계 및 안전성 분석을 하며, Ⅴ장에서는 제안 기법 구현을 통해 효율성을 분석한다. 마지막으로 Ⅵ장에서는 결론을 맺는다.

Ⅱ. 시스템 모델

본 장에서는 제안 기법의 시스템 모델(Fig. 1.)에 대하여 설명한다. 제안하는 기법을 수행하는 객체는 클라우드 서버, 키 서버, 그리고 클라이언트로 구성되며 역할은 다음과 같다.

2.1 클라우드 서버(Cloud Server)

클라우드 서버는 다수의 클라이언트들에게 스토리지 서비스를 제공한다. 클라우드 서버는 CPU 연산, 데이터에 접근하고 이를 메모리에 저장하는 I/O 연산 및 네트워크 대역폭 등 연산 능력이 클라이언트보다 훨씬 뛰어나다. 기본적으로 클라우드 서버는 신뢰할 수 없으나 서비스 이용에 필요한 연산을 정직하게 수행하고, 저장되어 있는 데이터에 대한 무결성(Integrity)을 보장(Honest-but-curious)한다고 가정한다.

2.2 키 서버(Key Server)

키 서버는 클라이언트가 클라우드 서버에 저장할

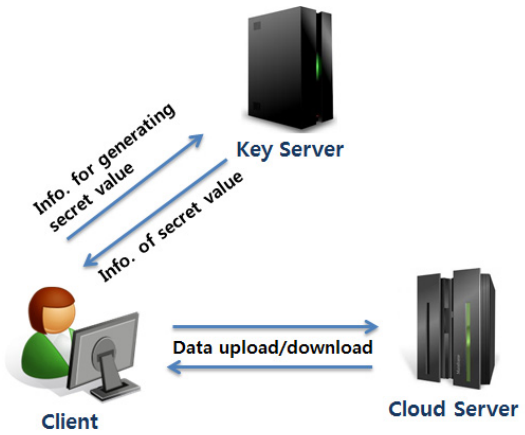

Fig. 1. Architecture of Proposed Scheme

데이터를 암호화하기 위해 필요한 비밀키에 대한 비밀값을 생성한다. 전수 조사 공격을 막기 위해 클라이언트마다 일정 기간 동안 데이터의 비밀키 질의를 할 수 있는 횟수를 제한하는 정책[6]을 설정하여 제한된 횟수 이상 비밀키 질의를 했을 경우에는 응답을 멈추도록 설정된다. 추가적으로, 키 서버는 클라이언트와의 통신에서 필요한 연산을 정직하게 수행한다.

2.3 클라이언트(Client)

클라이언트는 키 서버와의 통신에서 얻은 비밀 값을 이용하여 데이터에 대한 비밀키를 생성한 뒤 데이터를 암호화하고 태그 값을 생성한다. 생성한 태그 값을 이용하여 데이터의 저장여부를 확인하고, 만약 클라우드 서버에 데이터가 없다면 업로드 과정을 진행한다. 이후 클라우드 서버에 저장된 데이터가 필요한 경우, 클라이언트는 데이터 소유권 증명을 통해 클라우드 서버로부터 데이터를 다운로드 받을 수 있다.

III. 공격 모델

본 장에서는 클라이언트 사이드 중복 제거 기술이 적용된 클라우드 스토리지 환경에서 발생할 수 있는 공격 모델에 대하여 설명한다.

3.1 데이터 프라이버시(Data Privacy)

클라이언트 사이드 중복 제거 기술은 클라이언트가 태그 값을 이용해 클라우드 서버로부터 다운로드 권한을 부여 받기 때문에 권한이 없는 클라이언트가 데이터를 다운로드 하거나, 저장되어 있는 데이터가 암호화 되어 있지 않다면 데이터에 대한 정보가 노출되는 프라이버시 문제가 발생한다[2,6,7]. 공격자는 클라이언트와 클라우드 서버 간의 통신으로부터 받는 정보를 통해 저장된 데이터에 대한 어떠한 정보도 얻을 수 없어야 하고, 클라우드 서버 역시 저장된 데이터를 통해 데이터와 클라이언트의 어떠한 정보도 얻을 수 없어야 한다. DropBox의 경우, 클라이언트가 저장한 데이터들을 AES-256 암호화를 통해 안전하게 보호한다고 설명 하고 있다[1]. 하지만 암호화 키는 DropBox에서 생성하고 관리하기 때문에, 클라이언트가 저장한 데이터들이 암호화 되어 있지 않다면 데이터의 기밀성(Confidentiality)을 보장하지 못한다. 뿐만 아니라, DropBox 클라이언트의 계정 해킹 사고 사례[13]에서 만약 공격자가 다수의 클라이언트들의 데이터를 다운로드 했다면 그 피해는 더욱 커졌을 것이다.

3.2 전수 조사 공격(Brute-force Attack)

데이터의 안전성을 제공하기 위해 데이터를 암호화할 때, 확률적(Probablistic) 암호 기법을 사용한다면 동일한 데이터를 가진 클라이언트들이라 하더라도 생성하는 암호문이 서로 다르기 때문에 중복 제거 기술을 적용할 수 없다. 따라서 암호문에 대한 중복 제거 기술을 적용하기 위해서는 결정적(Deterministic) 암호 기법을 사용하여 동일한 데이터로부터 생성되는 암호문은 같아지도록 만들어야 한다. 하지만, 결정적 암호 기법을 사용하더라도 각각의 클라이언트마다 서로 다른 비밀키를 이용하여 데이터를 암호화하면, 암호문이 달라지기 때문에 여전히 중복 제거 기술을 적용할 수 없는 문제가 발생한다. 따라서 암호문에 대한 중복 제거 기술을 적용하기 위해서는, 동일한 데이터를 가진 클라이언트들은 동일한 키를 이용해 결정적 암호 기법을 사용하여 암호문을 생성해야 한다. 대부분의 클라이언트 사이드 중복 제거 기술의 경우, [3]에서 제시한 데이터를 해쉬한 값을 비밀키로 이용하여 데이터를 암호화하는데, 이는 공격자가 엔트로피가 낮거나 작은 도메인을 가지는 데이터들의 전수 조사 공격을 통해 데이터에 대한 비밀키를 생성할 수 있는 문제가 발생한다.

1) https://www.dropbox.com/

3.3 온라인 추측 공격(Online-guessing Attack)

클라이언트 사이드 중복 제거 기술은 클라우드 서버에 데이터의 저장 여부를 확인하는 태그가 크기가 작고 데이터로부터 결정적으로 생성되기 때문에 공격자가 클라이언트들이 저장한 데이터의 내용을 추측하여 알아낼 수 있는 문제점이 발생한다[1]. Fig. 2.와 같이 공격자가 엔트로피가 낮거나 작은 도메인을 가지는 목표 데이터에 대해, 데이터의 정보를 조금씩 변화시키면서 생성한 태그를 이용하여 클라우드 서버에 데이터가 저장되어 있는지의 여부를 알 수 있는데 이를 온라인 추측 공격이라 한다. 예를 들어, Alice와 Bob은 같은 회사의 같은 부서에서 근무하는 사원이라고 했을 때, Bob은 Alice의 개인 프로필에 작성된 Alice의 연봉을 알고 싶어 한다. 연봉은 노출되기 꺼려하는 민감한 정보라고 할 수 있다. 만약 Alice가 작성한 프로필을 클라우드 서버에 저장했다면, Bob은 일정 형식이 갖춰져 있는 프로필에 Alice의 정보를 입력한 후, 연봉을 바꿔가면서 태그 값을 생성하여 클라우드 서버에 저장되어 있는지 여부를 확인함으로써 Alice의 연봉을 알 수 있다. 이처럼 온라인 추측 공격은 공격자가 데이터 저장여부 뿐만 아니라 데이터의 정보까지 노출되기 때문에 데이터 프라이버시 문제도 함께 발생한다.

Fig. 2. Online-Guessing Attack

3.4 포이즌 공격(Poison Attack)

클라이언트 사이드 중복 제거 기술의 경우 클라이언트는 태그 값을 이용하여 데이터가 클라우드 서버에 저장되어 있는지 여부를 확인한다. 만약 클라이언트가 클라우드 서버에 저장되어 있는 데이터에 대해 무결성을 확인할 수 있는 방법이 없다면, 악의적인 클라이언트에 의해 변형된 데이터가 저장되어 있을 때 이후 데이터를 다운로드 시 원본 데이터를 잃어버리게 되는 문제가 발생하는데 이를 포이즌 공격이라 한다. 예를 들어, 악의적인 클라이언트 Bob이 원본 데이터 대신 동일한 크기의 변형된 데이터를 태그 값과 함께 저장

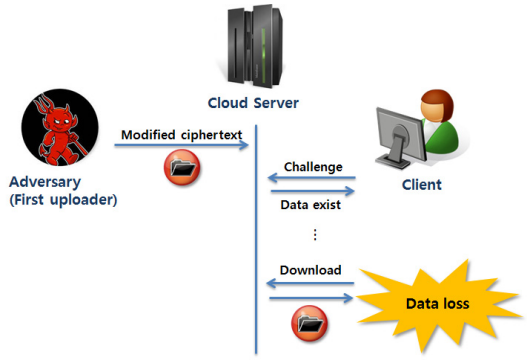

Fig. 3. Poison Attack

한다면, 이후에 동일한 데이터를 가지고 있는 클라이언트 Alice가 데이터를 업로드 하기 위해 태그 값을 클라우드 서버에 전송했을 때, Alice는 데이터가 이미 저장되어 있다는 메시지를 받게 된다. 데이터가 저장되어 있다는 메시지를 받은 Alice는 데이터를 업로드 하지 않고 지우게 되고 태그 값만 저장하는데, 이후 클라우드 서버에게 데이터 다운로드를 요청했을 때 변형된 데이터를 전송받게 되어 원본 데이터를 잃어버리게 된다(Fig. 3). 이처럼 데이터의 손실을 가져오는 포이즌 공격은 클라우드 스토리지를 사용하는 다수의 클라이언트뿐만 아니라 스토리지 서비스를 제공하는 클라우드 서버에게도 큰 피해를 발생시킨다.

IV. 제안 기법 설계 및 안전성 분석

본 장에서는 제안 기법 설계에 필요한 배경지식을 설명하고, Ⅲ장에서 제시한 공격에 안전한 클라이언트 사이드 중복 제거 기술을 설계한 후 안전성을 분석한다.

4.1 배경지식

4.1.1 RSA 블라인드 서명(RSA Blind Signature)

RSA 블라인드 서명 기법은 공개키와 비밀키를 생성하기 위한 키 생성 알고리즘 KeyGen, 서명 프로토콜 Sign Protocol과 검증 알고리즘 Ver으로 이루어져 있다. 해쉬 함수 $H:\{0,1\}^{*}\rightarrow Z_N$가 주어졌다고 가정한다.

- KeyGen(λ) : 서명자는 보안 상수 λ를 입력받아 공개키 $pk=(e,N)$와 비밀키 $sk=(N,d)$를 생성한다.

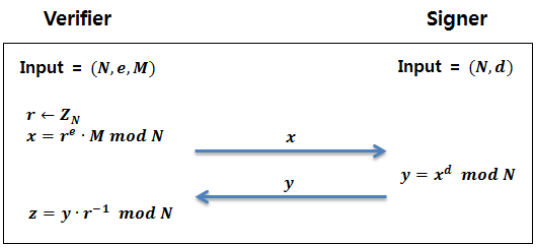

Fig. 4. RSA Blind Signature Protocol

- Sign Protocol

서명 프로토콜은 다음 Fig. 4.와 같다.

검증자는 서명자의 공개키 $pk=(e,N)$와 메시지 M을 입력받아 난수 $r \in Z_N$을 생성하여 $x=r^e \cdot M \bmod N$를 계산하여 서명자에게 전송한다. 서명자는 자신의 비밀키 $sk=(N,d)$를 이용하여 $y=x^d \bmod N$를 계산한 후 검증자에게 전송하면, 검증자는 난수 r의 역수를 계산하여 $z=y \cdot r^{-1} \bmod N$을 얻는다.

- $\mathrm{Ver}(pk, z, M)$: 검증자는 $z^e=M \bmod N$이 성립하는지 검증한다.

4.1.2 암호학적 해쉬 함수(Cryptographic Hash Function)

암호학적 해쉬 함수는 역상 저항성, 제 2 역상 저항성과 충돌 저항성의 성질을 만족한다.

- 역상 저항성(Preimage Resistance) : 해쉬값 y가 주어졌을 때, $h(x)=y$를 만족하는 x 값을 찾는 것이 계산적으로 불가능하다. 즉, 일방향성(One-Wayness)을 만족해야 한다.
- 제2 역상 저항성(Second Preimage Resistance) : $(x,h(x))$가 주어졌을 때, $h(x)=h(x'), x \neq x'$을 만족하는 x' 값을 찾는 것이 계산적으로 불가능하다.
- 충돌 저항성(Collision Resistance) : $h(x)=h(x'), x \neq x'$을 만족하는 (x,x')쌍을 찾는 것이 계산적으로 불가능하다. 이 때, (x,x')을 충돌 쌍(Collision Pair)이라 한다.

4.2 제안 기법

제안 기법은 클라이언트가 데이터를 암호화하기 위해 키 서버로부터 비밀 값을 얻어 비밀키를 생성하는 비밀키 생성 단계, 생성된 비밀키를 이용하여 데이터를 암호화 하고 태그 값을 생성하여 클라우드 서버에 데이터 업로드 요청을 하는 데이터 업로드 단계와 이후 클라우드 서버에 저장된 데이터를 다운로드 요청하는 데이터 다운로드 단계로 구성된다. Table 1.은 제안 기법에서 사용하는 표기법에 대한 설명이다.

Table 1. Notations used in the Proposed Scheme

notations	description
H, G	hash fuctions
H_1	the cryptographic hash fuction
pk_{KS}	public key of the key server
sk_{KS}	private key of the key server
$Enc_k(\cdot)$	symmetric encryption by using the secret key k
ID_c	the identifier of the client
sk_c	private key of the client
f	data
sk_f	secret key of the data f
T	the tag value by hashing sk_f
C	the ciphertext of the data f by using sk_f
C'	the ciphertext stored in the cloud server realted to the tag value T
K_c	the encrypted value of sk_f by using sk_c
h	the poison attack verifier generated by the cloud server
h'	the data ownership value generated by the client
s, k	random values

4.2.1 비밀키 생성 단계

두 개의 해쉬 함수 $H: \{0,1\}^* \rightarrow Z_N$, $G: Z_N \rightarrow \{0,1\}^k$와 데이터의 비밀키 공간 $\{0,1\}^k$가 주어져 있고, 키 서버의 공개키와 비밀키는 각각 RSA 블라인드 서명 기법 Fig. 4의 공개키 $pk_{KS}=(e,N)$ 와 비밀키 $sk_{KS}=(N,d)$ 로 설정한다.

- 클라이언트는 데이터 f의 해쉬 값 $h=H(f)$을

계산하고 난수 $r \in Z_N$을 생성하여 $x = r^e \cdot h \bmod N$를 키 서버에 전송 한다.

- 키 서버는 자신의 비밀키를 이용하여 클라이언트에게 $y = x^d \bmod N$를 전송한다.
- 클라이언트는 키 서버로부터 받은 값 y를 이용하여 $z = y \cdot r^{-1} \bmod N$를 계산한 후, $z^e = h \bmod N$ 을 만족하면 데이터 f의 비밀키 $sk_f = G(z)$ 를 계산하고, 그렇지 않으면 $\perp$를 출력한다.

4.2.2 데이터 업로드 단계

암호학적 해쉬 함수 $H_1 : \{0,1\}^* \rightarrow \{0,1\}^l$가 주어져 있고, 데이터 암호화 $Enc_{sk}(\cdot)$는 안전한 대칭키 암호 기법을 사용한다고 가정한다.

4.2.2.1 데이터가 저장되어 있는 경우

데이터가 이미 저장되어 있는 경우, 클라우드 서버는 중복 제거 기술을 이용하여 데이터를 중복 저장하지 않는다. 과정은 다음 Fig. 5.와 같다.

- 클라이언트는 키 서버와의 통신으로부터 생성한 데이터 f의 비밀키 sk_f를 해쉬하여 태그 값 $T = H_1(sk_f)$을 생성하고, 난수 $s \in \{0,1\}^*$를 선택하여 데이터 업로드 요청 메시지와 함께 태그 값 T과 난수 s를 클라우드 서버에 전송한다.
- 클라우드 서버는 클라이언트로부터 전송받은 태그 값 T와 일치하는 태그 값이 있는지 검색한다. 일치하는 값이 있다면, 태그 값 T에 저장되어 있는 암호문 C'을 이용하여 $h = H_1(C' \| s)$ 값을 계산하고, 난수 $k \in \{0,1\}^*$를 선택하여 클라이언트에게 h 값과 k를 전송한다.
- 클라우드 서버로부터 h 값과 난수 k를 전송받았다면, 클라이언트는 데이터 f의 암호문 $C = Enc_{sk_f}(f)$을 생성하고, 생성한 암호문 C과 난수 s를 이용하여 $H_1(C \| s)$를 계산한 후 $h = H_1(C \| s)$가 성립하는지 확인하여 클라우드 서버에 원하는 암호문이 저장되어 있음을 검증한다. 검증을 통과하면 $h' = H_1(C \| k)$와 암호화 키 $K_c = Enc_{sk_c}(sk_f)$를 생성하여 클라이언트의 식별자 ID_c와 함께 승인 메시지를 클라우드 서버에 전송하고, 검증을 통과하지 못하면 오류 메시지를 클라우드 서버에게 전송한다.
- 클라이언트로부터 승인 메시지와 함께 h', ID_c와 K_c를 전송 받았다면, 클라우드 서버는 저장되어 있는 암호문 C'와 난수 k을 이용하여 $H_1(C' \| k)$를 계산한 후 $h' = H_1(C' \| k)$이 성립하는지 확인하여 소유권 검증을 한다. 소유권 검증이 완료되면 클라이언트에게 승인 메시지를 전송하고, 클라이언트로부터 받은 암호문 C, 식별자 ID_c와 암호화 키 K_c를 태그 T와 함께 저장한다. 소유권 검증을 통과하지 못했다면, 오류 메시지를 클라이언트에게 전송한다.

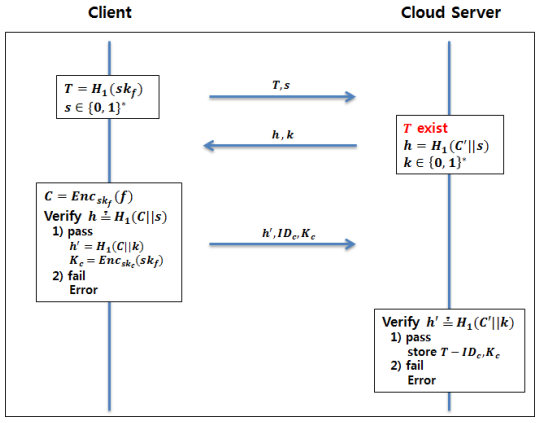

Fig. 5. Data Upload Process when the Data was already stored in the Cloud Server

4.2.2.2 데이터가 저장되어 있지 않은 경우

데이터가 저장되어 있지 않은 경우, 클라우드 서버는 클라이언트에게 데이터 업로드 요청을 한다. 과정은 다음 Fig. 6.와 같다.

- 클라이언트는 업로드 요청 메시지와 함께 태그 값 T과 난수 s를 클라우드 서버에 전송한다.
- 클라이언트로부터 전송받은 태그 값 T과 일치하는 값이 없다면, 클라우드 서버는 데이터 업로드 요청 메시지를 클라이언트에게 전송한다.
- 클라우드 서버로부터 데이터 업로드 요청을 받았다면, 클라이언트는 생성한 암호문 C, 클라이언트의

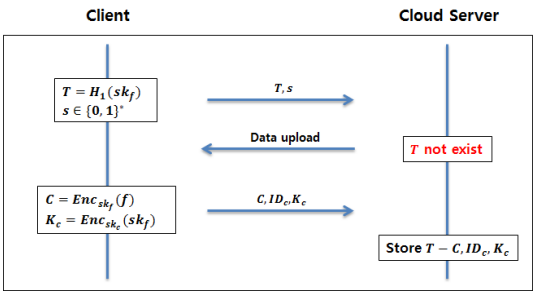

Fig. 6. Data Upload Process when the Data was not stored in the Cloud Server

식별자 ID_c, 암호화 키 K_c와 h 값을 클라우드 서버에 전송한다.

- 클라우드 서버는 전송받은 암호문 C, 클라이언트의 식별자 ID_c와 암호화 키 K_c를 태그 값 T과 함께 저장한다.

4.2.3 데이터 다운로드 단계

- 클라이언트는 데이터 다운로드 요청 메시지와 함께 태그 값 T과 클라이언트 식별자 ID_c를 클라우드 서버에 전송한다.
- 클라우드 서버는 클라이언트에 대한 데이터 소유권 검사를 실행한 뒤, 검증이 완료 되었다면 태그 값 T을 이용하여 클라이언트 식별자 ID_c와 함께 저장되어 있는 암호문 C과 암호화 키 K_c를 클라이언트에게 전송한다.
- 클라이언트는 클라우드 서버로부터 받은 암호화 키 K_c를 복호화 하여 데이터 f의 비밀키 sk_f를 생성하고, 이를 이용하여 암호문 C을 복호화 하여 데이터 f를 얻는다.

4.3 안전성 분석

본 절에서는 클라이언트 사이드 중복 제거 기술에서 발생할 수 있는 보안 취약점에 대해 각각의 공격 모델을 바탕으로 제안 기법의 안전성을 분석한다.

4.3.1 데이터 프라이버시

클라이언트 사이드 중복 제거 기술의 경우, 공격자는 클라우드 서버에 저장된 데이터에 대한 접근 권한을 얻기 위해 태그 값 T을 이용하여 클라우드 서버와 질의-응답 메커니즘을 수행한다. 이때, 클라우드 서버가 태그 값을 가진 누구에게나 다운로드 권한을 부여하거나, 클라우드 서버에 저장되어 있는 데이터가 암호화가 되어 있지 않다면, 다운로드 권한을 얻은 공격자는 저장되어 있는 데이터에 대한 정보를 얻을 수 있게 되기 때문에 데이터에 대한 프라이버시 문제가 발생하게 된다.

본 논문의 제안 기법에서 클라이언트는 클라우드 서버에 저장할 데이터를 암호화하기 위해 키 서버와의 통신에서 얻은 비밀 값을 이용하여 비밀키 sk_f를 생성한 후, 이를 이용하여 데이터 f를 암호화한 암호문 $C = Enc_{sk_f}(f)$을 생성하여 저장한다. 만약 공격자가 태그 값 T을 얻어 클라우드 서버와 질의-응답 메커니즘을 수행할 경우, 일차적으로 소유권에 대한 검증을 할 수 없기 때문에 클라우드 서버에 저장되어 있는 암호문에 대한 다운로드 권한을 얻지 못한다. 만약 공격자가 다운로드 권한을 획득하여 암호문 C와 암호화 키 K_c를 얻었다 하더라도, 클라이언트의 비밀키 sk_c로 암호화 된 K_c로부터 데이터 f의 비밀키 sk_f를 얻을 수 없기 때문에 암호문 C를 복호화 하는 것은 불가능하다. 따라서 본 논문의 제안 기법은 데이터 소유권 검증을 통해 다운로드 권한을 부여하고, 안전한 암호화 기법을 사용하여 데이터를 암호화하여 저장함으로써, 공격자에게 데이터 정보 노출을 막을 수 있기 때문에 데이터 프라이버시에 대한 문제를 해결할 수 있다.

4.3.2 전수 조사 공격

본 논문의 제안 기법에서는 목표 데이터 f에 대한 비밀키 sk_f를 얻기 위해 공격자는 키 서버와 통신을 통해 후보가 되는 데이터의 비밀 값을 얻어 비밀키를 생성하거나, 데이터 f를 소유한 클라이언트와 키 서버 간의 통신을 도청하여 얻은 정보를 이용하여 비밀키를 생성할 수 있다. 하지만 공격자가 목표 데이터 f의 비밀키 sk_f를 생성하기 위해 후보가 되는 데이터를 이용하여 키 서버에 반복적으로 통신을 요청하는 공격은, 키 서버의 클라이언트마다 일정 기간 동안 비밀키 질의를 할 수 있는 횟수를 제한하는 정책[6] 때문에 불가능하다. 또한, 공격자는 클라이언트와 키 서버 간 통신 도청을 통해 x값과 y값을 얻을 수 있다(Fig. 4). 하지만 RSA 알려진-타겟 인버전 공격 문제

Table 2. Security analysis of Our Proposed Scheme in Comparison with Other Client-side Deduplication Schemes

shcemes	data privacy	brute-force attack	online-guessing attack	poison attack
[1]	O	X	X	X
[5]	O	O	X	X
[9]	O	X	O	X
proposed scheme	O	O	O	O

(RSA Known-target Inversion Problem)가 여전히 어렵다면, 공격자는 도청을 통해 얻은 x와 y 값을 이용하여 클라이언트가 생성한 난수 r에 대한 정보를 얻을 수 없기 때문에 데이터의 비밀키 sk_f를 얻는 것은 쉽지 않다[4]. 따라서 본 논문의 제안 기법은 목표 데이터 f에 대해서 공격자가 비밀키 sk_f를 생성하는 전수 조사 공격을 효과적으로 막을 수 있다.

4.3.3 온라인 추측 공격

대부분의 클라이언트 사이드 중복 제거 기술에서 클라이언트는 데이터 업로드 시 태그 값을 전송하여 클라우드 서버에 데이터가 저장되어 있는지, 또는 저장되어 있지 않은지 확인한다. 이 때, 데이터를 해쉬한 값을 태그 값 $T=H(f)$으로 사용하는데, 이는 앞서 살펴본 전수 조사 공격에서와 같이 엔트로피가 낮거나 작은 도메인을 가지는 목표 데이터 f에 대해서 공격자가 후보가 되는 데이터의 태그 값을 생성하여 클라우드 서버에 전송함으로써, 목표 데이터 f가 클라우드 서버에 저장이 되어 있는지 여부를 알 수 있다.

본 논문의 제안 기법에서는 키 서버와의 통신으로부터 생성한 데이터에 대한 비밀키 sk_f를 해쉬하여 태그 값 $T=H(sk_f)$을 생성한다. 공격자가 목표 데이터 f의 저장 여부를 확인하기 위해서는, 후보가 되는 데이터의 태그 값을 생성해야 하며, 각각의 태그 값을 생성하기 위해서는 키 서버로부터 후보 데이터의 비밀키를 얻어야만 한다. 하지만 이는 일정 기간 동안 비밀키 생성 질의 횟수 제한을 하고 있는 키 서버 정책[6] 때문에 공격자가 후보가 되는 데이터의 비밀키를 모두 얻는 것은 불가능하고, 이에 태그 값을 모두 생성하는 것은 불가능하게 된다. 따라서 공격자는 클라우드 서버에 후보 데이터에 대한 태그 값 검색을 할 수 없고, 목표 데이터 f가 클라우드 서버에 저장되어 있는지의 여부를 알 수 없기 때문에 본 논문의 제안 기법은 온라인 추측 공격을 막을 수 있다.

4.3.4 포이즌 공격

지금까지 제안된 클라이언트 사이드 중복 제거 기술에서는 클라이언트가 클라우드 서버에 저장되어 있는 데이터에 대한 무결성을 효과적으로 확인할 수 있는 기법이 없다.

본 논문의 제안 기법에서 암호문은 결정적으로 생성되며, 데이터 업로드 시에 클라이언트는 난수를 생성하여 클라우드 서버에 전송한다. 클라우드 서버는 저장되어 있는 암호문과 클라이언트로부터 전송받은 난수를 이용하여 해쉬한 값을 다시 클라이언트에게 보내면, 클라이언트는 클라우드 서버로부터 받은 값을 검증한다. 만약 공격자(최초 업로더)가 클라우드 서버에 변형된 암호문을 저장했다면, 비록 동일한 태그 값이 저장되어 있다하더라도 검증 과정을 통과할 수 없기 때문에 클라이언트가 저장된 데이터가 올바르지 않다는 것을 알 수 있다. 따라서 본 논문의 제안 기법은 클라이언트는 저장된 데이터의 무결성에 대한 검증을 할 수 있기 때문에 데이터 손실을 가져오는 포이즌 공격을 막을 수 있다.

V. 구현 및 효율성 분석

본 장에서는 제안 기법을 구현하고 일반 데이터 업로드 수행 시간과의 비교를 통해 효율성을 분석한다. 제안 기법이 적용된 환경에서 클라이언트는 해쉬 및 데이터 암호화 연산 등을 추가적으로 수행해야 한다. 하지만 데이터의 저장 여부에 따라 일반 데이터 업로드 수행 시간과 비교했을 때, 데이터가 이미 저장되어 있을 경우 제안 기법이 적용됨으로써 업로드 시간 및 전송량을 줄일 수 있기 때문에 효율성을 제공하고, 데이터가 저장되어 있지 않을 경우에도 일반 데이터 업로드 과정과 수행 시간이 많이 차이나지 않음을 알 수 있다.

본 논문의 제안 기법을 구현하기 위해 충돌쌍 저항성을 가지는 암호학적 해쉬 함수 SHA256와 의미론적으로 안전한(Semantic Secure) 대칭키 암호화 기법 AES256(Advanced Encryption Standard)을 사용하였다. 모든 프로그램은 자바(Java)를 이용하여 구현 하였고, Intel(R) Core

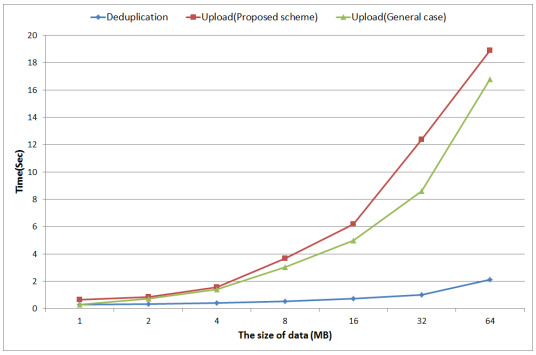

Fig. 7. Comparison between the Running Time of Our Protocol and that of General Data Upload Process without encryption

2 Duo 3GHz CPU, 4GB RAM의 PC 환경에서 실험하였다.

임의로 생성한 데이터의 크기별로 제안 기법의 데이터 업로드 단계와 일반 데이터(암호화되지 않은 데이터) 업로드 작업 수행 시간에 대해 측정하였으며, 각각의 결과 값은 100번의 반복된 실험을 통해 얻은 결과 값이다(데이터의 크기 : 2^iMB, $i = 0, \cdots, 6$). 각각의 데이터 크기별로 측정한 결과 값은 다음 Fig. 7.과 같다.

데이터가 클라우드 서버에 저장되어 있을 경우, 크기가 작은 데이터(1MB, 2MB)는 본 논문의 중복 제거 기술과 일반 데이터 업로드 수행 시간이 거의 차이나지 않는다. 하지만 데이터의 크기(4MB 이상)가 커질수록 중복 제거 기술의 수행 시간은 일반 데이터 업로드 수행 시간보다 훨씬 빠르게 진행되는 것을 알 수 있다. 예를 들어, 64MB 크기의 데이터의 경우, 중복 제거 기술의 수행 시간은 일반 업로드 수행 시간의 약 12.5%에 불과하다.

데이터가 클라우드 서버에 저장되어 있지 않을 경우, 본 논문의 제안 기법은 일반 데이터 업로드 과정과 비교했을 때, 데이터를 업로드 하기 전에 해쉬 연산과 데이터 암호화 연산(AES256)을 추가적으로 진행하게 된다. 하지만 해쉬 연산과 데이터 암호화 연산 속도는 매우 빠르기 때문에 본 논문의 제안 기법과 일반 데이터 업로드 수행 시간은 많이 차이나지 않는 것을 알 수 있다.

VI. 결 론

본 논문에서는 클라이언트 사이드 중복 제거 기술 환경에서 발생할 수 있는 다양한 공격으로부터 안전하고 효율적인 중복 제거 기술을 제안하였다. 일정 기간 동안 데이터의 비밀키 질의를 할 수 있는 횟수를 제한하는 정책을 가진 키 서버를 둠으로써 전수조사 공격 및 온라인 추측 공격에 대한 안전성을 제공한다. 또한, 데이터 업로드 시 클라이언트가 클라우드 서버에 저장된 암호문의 무결성을 검증함으로써 데이터 손실을 가져오는 포이즌 공격에 대한 안전성을 제공한다. 마지막으로, 데이터의 크기별로 제안 기법과 일반 데이터 업로드 과정의 수행 시간을 비교한 결과로부터 데이터의 크기가 커질수록 제안 기법이 높은 효율성을 제공하는 것을 확인할 수 있다.

References

[1] D. Harnik, B. Pinkas, and A. Shulman-Peleg, "Side channels in cloud services : Deduplication in cloud storage," IEEE Security and Privacy Magazine, vol.8, pp. 40-47, Nov. 2010.

[2] M. Green, S. Hohenberger, and B. Waters, "Outsourcing the decryption of abe ciphertexts," Proc. USENIX Conf. Security (SEC' 11), Aug. 2011.

[3] J.R. Douceur, A. Adya, W.J. Bolosky, D. Simon, and M. Theimer, "Reclaiming space from duplicate files in a serverless distributed file system." Proc. Int'l Conf. Distributed Computing Systems (ICDCS' 02), pp. 617-624, Jul. 2002.

[4] M. Bellare, C. Namprempre, D. Pointcheval, and M. Semanko. "The one-more-RSA-inversion problems and the security of Chaum's blind signature scheme." J. Cryptology, vol. 16, no.3, pp. 185-215, Jun. 2003.

[5] J. Xu, E. C. Chang, and J. Zhou. "Weak Leakage-Resilient client-side deduplication of encrypted data in cloud storage." ASIA CCS, pp. 195-206, May. 2013.

[6] M. Bellare, and S. Keelveedhi. "DupLess : Server-aided encryption for deduplicated storage." Proc. USENIX conf. Security (SEC' 13), pp. 179-194, Aug.

2013.

〔7〕 S. Halevi, D. Harnik, B. Pinkas, A. Shulman-Peleg, "Proofs of Ownership in Remote Storage Systems," Proc. 18th ACM. Conf. Computer and Communications Security (CCS '11), ACM Press, pp. 491-500, Oct. 2011.

〔8〕 D. Russell, "Data deduplication will be even bigger in 2010," Gartner, Feb. 2010.

〔9〕 Y. Shin, and K. Kim, "Efficient and Secure File Deduplication in Cloud Storage," IEICE TRANS. INF. & SYST., Vol. E97-D, NO.2, pp. 184-197, Feb. 2014.

〔10〕 D. Boneh, G.D. Crescenzo, R. Ostrovsky, and G. Persiano, "Public Key Encryption with Keyword Search," In Proceedings of Eurocrypt '04, LNCS Vol. 3027, pp. 506-522, Springer-Verlag, May. 2004.

〔11〕 K. Park, J. E. Eom, D. H. Lee, "An analysis of Shin et al's scheme for secure deduplication," Korea Institute of Information Security & Cryptology CISC-S, pp. 50, Jun, 2014.

〔12〕 M. Dutch and L. Freeman. "Understanding data deduplication ratios." SNIA Data Management Forum, Jun. 2008.

〔13〕 "DropBox, Hacked on client accounts", http://www.bloter.net/archives/121266

〈저 자 소 개〉

박 경 수 (Kyungsu Park) 학생회원
2012년 8월: 숭실대학교 수학과 학사 졸업
2013년 8월~현재: 고려대학교 정보보호대학원 석사과정
〈관심분야〉 암호프로토콜, 암호이론, 클라우드 컴퓨팅

엄 지 은 (Ji Eun Eom) 학생회원
2010년 2월: 고려대학교 수학과 학사 졸업
2012년 2월: 고려대학교 정보보호대학원 석사 졸업
2013년 3월~현재: 고려대학교 정보보호대학원 박사과정
〈관심분야〉 암호프로토콜, 암호이론, 클라우드 컴퓨팅, 인증 및 키 교환

박 정 수 (Jeongsu Park) 학생회원
2002년 2월: 고려대학교 전산학과 학사 졸업
2006년 7월: 고려대학교 정보보호대학원 석사 졸업
2012년 3월~현재: 고려대학교 정보보호대학원 박사과정
〈관심분야〉 정보보호, 모바일 보안, MPC

이 동 훈 (Dong Hoon Lee) 종신회원
1983년 8월: 고려대학교 경제학과 학사 졸업
1987년 12월: Oklahoma University 전산학과 석사 졸업
1992년 5월: Oklahoma University 전산학과 박사 졸업
1993년 3월~1997년 2월: 고려대학교 전산학과 조교수
1997년 3월~2001년 2월: 고려대학교 전산학과 부교수
2001년 3월~현재: 고려대학교 정보보호대학원 교수
〈관심분야〉 암호프로토콜, 암호이론, USN이론, 키 교환, 익명성 연구, PET 기술