Secure Data Service Outsourcing with Untrusted Cloud

Huijun Xiong

Dissertation submitted to the Faculty of the

Virginia Polytechnic Institute and State University

in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

in

Computer Science

Danfeng Yao, Chair

Dennis G Kafura

Christopher L North

Wenjing Lou

Xinwen Zhang

April 29, 2013

Blacksburg, Virginia

Keywords: Cloud Computing, Outsource Data Security, Proxy Re-encryption, Content Delivery Network, Key Management

Secure Data Service Outsourcing with Untrusted Cloud

Huijun Xiong

ABSTRACT

Outsourcing data services to the cloud is a nature fit for cloud usage. However, increasing security and privacy concerns from both enterprises and individuals on their outsourced data inhibit this trend. In this dissertation, we introduce service-centric solutions to address two types of security threats existing in the current cloud environments: semi-honest cloud providers and malicious cloud customers. Our solution aims not only to provide confidentiality and access controllability of outsourced data with strong cryptographic guarantee, but, more importantly, to fulfill specific security requirements from different cloud services with effective systematic ways.

To provide strong cryptographic guarantee to outsourced data, we study the generic security problem caused by semi-honest cloud providers and introduce a novel proxy-based secure data outsourcing scheme. Specifically, our scheme improves the efficiency of traditional proxy re-encryption algorithm by integrating symmetric encryption and proxy re-encryption algorithms. With less computation cost on applying re-encryption operation directly on the encrypted data, our scheme allows flexible and efficient user revocation without revealing underlying data and heavy computation in the untrusted cloud.

To address specific requirement from different cloud services, we investigate two specific cloud

services: cloud-based content delivery service and cloud-based data processing service. For the former one, we focus on preserving cache property in the content delivery network and propose *CloudSeal*, a scheme for securely and flexibly sharing and distributing content via the public cloud. With the ability of caching the major part of a stored cipher content object in the delivery network for content distribution and keeping the minor part with the data owner for content authorization, CloudSeal achieves security and efficiency both theoretically and experimentally. For the later service, we design and realize *CloudSafe*, a framework that supports secure and efficient data processing with minimum key leakage in the vulnerable cloud virtualization environment. Through the adoption of one-time cryptographic key strategy and a centralized key management framework, CloudSafe efficiently avoids cross-VM side channel attack from malicious cloud customers in the cloud. Our experimental results confirm the practicality and scalability of CloudSafe.

To my beloved husband Guanying Wang and my parents, who are always there for me.

ACKNOWLEDGEMENTS

This dissertation was completed with a lot of help from my academic advisor, my mentor, my family, and my friends. First and foremost, I would like to thank my academic advisor, Dr. Danfeng Yao. for her insightful advice, patient instruction, and constant support during my five-year graduate study. Particularly, I am grateful for her understanding and help during my pregnancy and the first three month after the labor. Without her support, it is impossible for me to finish my Ph.D study.

I want to give my special thanks to Dr. Xinwen Zhang for his guidance during my thesis. He introduced me to the topic, discussed with me about the problem, and worked with me on the paper. I felt so lucky to be guided by him during my internship at Huawei and to be able to continue working with him after i was back to school. I am also grateful to have Dr. Dennis G. Kafura, Dr. Wenjing Lou, Dr. Chris North as my other committees. They devoted considerable time and effort on my thesis, slides, and presentations. Their valuable suggestions and comments greatly improve my final thesis.

I give my deepest appreciation to my family for their never-ending support during my study. My husband, Guanying Wang, was constantly encouraging me to pursue my Ph.D degree and took care of me whenever he could. My mom always put my needs before hers, especially when she was visiting me here. My daughter, Lana Wang, was my sweetheart who taught me unconditional love and helped me build the *now* habit.

I also want to thank my friends at Virginia Tech. Kui Xu, Hussain Almohri, Xiaokui Shu, Hao

Zhang, Karim Elish, and Fang Liu are my dearest lab mates, they made good companion to me when I was at lab. Chunyi Su and Min Li are great researchers on system, they helped me a lot when I was doing experiments. Shucai Xiao and Heshan Lin are like big brothers, I enjoyed my conversation with them.

Contents

1 Introduction **1**

1.1 Research Challenges 2

1.2 Research Contributions 4

1.2.1 Proxy-based Secure Data Outsourcing with Public Cloud 5

1.2.2 Secure Cloud-based Content Delivery Service with Proxy Re-encryption . 7

1.2.3 Secure Cloud-based Data Processing Service with Vulnerable Cloud Virtualization Environment 9

1.3 RoadMap 11

2 Literature Review and Preliminary Techniques **13**

2.1 Literature Review 13

2.1.1 Security in Cloud Computing 14

2.1.2 Outsourced Data Security 19

2.1.3 Security in Content Delivery 21

2.1.4 Side-channel Attack and Its Countermeasures 22

2.2 Preliminary Techniques 23

2.2.1 Proxy Re-encryption 24

2.2.2 Auxiliary Techniques 26

2.2.3 Complexity Assumption 26

3 Proxy-based Secure Data Outsourcing with Public Cloud 28

3.1 Problem Description and Motivation 28

3.2 The Scheme 30

3.2.1 Algorithm Definitions 30

3.2.2 Main Construction 33

3.2.3 Usage Protocols 36

3.2.4 Security Analysis 38

3.3 Implementation and Evaluation 40

3.3.1 Implementation 40

3.3.2 Evaluation 41

3.4 Chapter Summary 41

4 Secure Cloud-based Content Delivery Service with Proxy Re-encryption 44

4.1 Problem Description and Motivation 44

4.2 Models and Design Goals 46

4.2.1 The Scenario 46

4.2.2 Security and System Goals 48

4.3 CloudSeal Scheme Details 52

4.3.1 Overview 52

4.3.2 User Management 55

4.3.3 Security Analysis 57

4.4 Implementation and Evaluation 59

4.4.1 Implementation 59

4.4.2 Experimental Evaluation 62

4.5 Chapter Summary 70

5 Secure Data Processing Service with Vulnerable Cloud Virtualization Environment 71

5.1 Problem Description and Motivation 72

5.2 Models and Design Goals 77
5.2.1 Problem Scenario 77
5.2.2 Threat Model 80
5.3 Design Overview 82
5.3.1 Initial Attempts 82
5.3.2 Design Strategies 83
5.3.3 CloudSafe Overview 85
5.4 Cryptographic Scheme 90
5.4.1 Cryptographic Scheme for CloudSafe 90
5.4.2 Construction and Discussion 91
5.5 Key Management 94
5.5.1 Overview 95
5.5.2 Secure Key Distribution 97
5.5.3 Virtual Server Lifecycle and Authentication 99
5.5.4 Discussion 101
5.6 Implementation & Evaluation 101
5.6.1 Implementation 101

5.6.2 Evaulation 103

5.7 Chapter Summary 107

6 Conclusion and Future Work **108**

6.1 Conclusion 108

6.2 Future Work 110

6.2.1 Secure Cloud Collaboration Services 110

6.2.2 Secure Live Migration in the Cloud 111

Bibliography **113**

This page intentionally left blank

List of Figures

3.1 Protocols of our scheme used in the cloud 37

3.2 Efficiency comparison between proxy re-encryption scheme proposed in [28] and standard AES algorithm. 42

3.3 Efficiency comparison of encryption operation between our scheme and standard AES algorithm. 42

3.4 Efficiency comparison of decryption operation between our scheme and standard AES algorithm. 43

4.1 Schematic drawing of data flow in cloud-based storage and delivery services. . . . 47

4.2 CloudSeal overview. 53

4.3 CloudSeal content delivery strategy 55

4.4 Overhead of encryption operations with different pairing types. 64

4.5 Overhead of decryption operations with different pairing types. 64

4.6 Number of required re-encryption operations with different churning rates 67

4.7 Downloading time for content delivery with storage center at Tokyo. 68

4.8 Downloading time for content delivery with storage center at N. California. 68

5.1 The problem scenario of CloudSafe 79

5.2 CloudSafe workflow 86

5.3 The overlay of key management framework 95

5.4 Cascade structure of the global key storage. 96

5.5 Key distribution protocol with CloudSafe. 99

5.6 The implementation of the key server with NFS. 103

5.7 The implementation of a key client with NFS client. 103

5.8 Comparative results of proxy re-encryption operations on different sizes of encrypted files. 104

5.9 Network latency of different key distribution scenarios. 106

This page intentionally left blank

List of Tables

3.1 Executing time of operations in proxy re-encryption in [28]. The algorithm is implemented with `pbc` library [12] with `TYPE E` elliptic curve on a desktop with Intel Duo CPU 2.93GHz 4GB RAM CentOS 2.6. 31

4.1 Re-encryption Time (Seconds) of 600MB Content on Different Amazon EC2 Instances. 66

This page intentionally left blank

Chapter 1

Introduction

Nowadays, cloud computing has penetrated into every corner of Internet industry with its low-cost computing resources, easy scaling architectures, and everywhere on-demand services. Currently, there are three most popular cloud service models: Software-as-a-Service (SaaS), Platform-as-a-Service (PaaS), and Infrastructure-as-a-Service (IaaS). SaaS provides application software through network, such as document processing software; PaaS offers web-based development environment usually with pre-installed operating systems, compilers, and database; IaaS delivers virtual machines, storage, servers, and so on with on-demand feature. A cloud provider can provide a specific cloud service or a combination of services to their customers. For example, Amazon [3] provides various IaaS services. Microsoft [9] sells all the three types of services. Customers can obtain cloud services by public cloud hosted by a third party cloud provider, or private cloud owned by themselves, or community cloud shared by different organizations, or hybrid cloud with several cloud bounded together.

In this chapter, we present the research challenges and contributions of this dissertation.

1.1 Research Challenges

Recently individuals and enterprises are hesitant to adopt public cloud services due to security and privacy concerns on their outsourced data in the cloud. Their hesitances are reasonable. On one hand, people want to take fully advantage of storage and computational resources from the cloud to store and process their data; on the other hand, people want to still maintain fully control over their outsourced data even when they do not have physical possession of them. So far, few cloud providers have provided security support for their customers. Instead, cloud customers are responsible to provide the security guarantee for their own outsourced data in the cloud. The most likely reason is the trade off between system performance and security mechanisms no matter how important security is for the system. We can not ignore these security problems any longer as the rising concerns about data security and privacy issues from cloud customers will jeopardize the future of cloud computing industry. It is urgent to develop efficient yet secure cloud services to eliminate these concerns.

In this dissertation, we address two types of security threats existing in the current public cloud environment: semi-honest cloud providers and malicious cloud customers. Lack of trust relationship between the cloud providers and cloud customers makes it hard for the customers to fully trust cloud providers when they outsource their data to the cloud. Evidencing by recent security researches, malicious cloud customers are able to steal secrecy from their neighbor cloud customers

with the newly discovered vulnerabilities in the cloud virtualization environment [108, 46, 31, 2].

Designing effective cryptographic techniques seems to be a promising direction to alleviate those security issues. Researchers have already proposed many solutions. For example, FHE (fully homophobic encryption) allows computation to be applied directly on encrypted data and returns the same results as that on plaintext of the data. Since the data always stays in its encrypted form when rest at the other parties, the data owner no longer needs to worry about the security of her outsourced data. However, due to its computing limitation and complexion, the practicality of FHE is still not clear now. In order to utilize computing resources in the public cloud, we need to work on the plaintext of the outsourced encrypted data in the public cloud environment. FDE (fully disk encryption) is another type of solution, which relies on encryption algorithms and key management mechanisms to ensure the confidentiality and access control to the outsourced data. This technique is able to solve the data security problem in the cloud to some extent. However, those cryptography-centric solutions, besides unsatisfied performance in the large-scale environment, are threatened by the attacks from malicious cloud customers when the victim cloud customer wants to use the cryptographic keys to decrypt her data in the cloud virtualization environment. This technique also fails in meeting unique system-related features of some cloud services, such as optimization strategies that are used in cloud-based content delivery services, which is not efficient enough for practical usage.

This dissertation addresses the security issues in current cloud computing industry with focus on security problems existing in various data services that have been outsourced to the cloud. Not only are cryptographic techniques explored to help data owners enforce data access control in the

cloud, but also systematic methods are studied to provide efficient data protection with untrusted virtualization environment in the cloud and different cloud services. Specifically, we utilize cryptographic means to protect data confidentiality and access controllability in the public cloud. We also explore effective systematic mechanisms to fulfill specific security requirements from different types of cloud services.

1.2 Research Contributions

In response to the mentioned security issues in the current cloud computing industry, we propose service-centric security solutions as a new direction to develop efficient security solutions. The advantage of service-centric solutions lies in two part. For one part, service-centric solutions are comprehensive solutions. They not only focus on cryptographic techniques to protect cloud customers from semi-honest cloud providers, but also explore systematic means to address security issues existing in the vulnerable cloud infrastructure. Having an open and large computing environment as the public cloud, we believe more and more efforts are needed to be made to securely use public cloud infrastructure in the future. For the other part, service-centric solutions are practical-oriented solutions. Instead of developing a generic solution for the entire cloud, service-centric solution focus on each distinct cloud services which has different security requirement and demand advanced system performance. This extra emphasis makes service-centric solution more suitable for real cloud usage.

Specifically, we present our service-centric solution in three aspects. We firstly propose a proxy-

based data outsourcing scheme to support efficient and secure data outsourcing operations for the data owner. This scheme handles the security issues caused by semi-honest cloud providers which is not trusted by most cloud services. Then, we adjust this scheme by studying the unique security and system requirements from two different cloud services: cloud-based content delivery service where the data is only stored and distributed over the public cloud and cloud-based data processing service where the data is processed in the public cloud environment. For the former one, we design and develop a new content delivery strategy to prompt the usage of our cryptographic scheme while preserving the efficiency of the original cloud-based content delivery services; for the later one, we integrate our cryptographic scheme with a novel key management scheme so that the cryptographic scheme is secure enough to be applied within a vulnerable virtualization environment in the public cloud;

1.2.1 Proxy-based Secure Data Outsourcing with Public Cloud

Outsourcing data to an untrusted server has been studied for decades. Researchers have proposed many solutions to protect confidentiality and provide access control to the outsourced data. Among them, proxy-based re-encryption scheme designed by Ateniese [28] is an advanced model known for all of its achieved properties. In this model, a proxy, controlled by the data owner, is usually sitting between the untrusted environment and data users. Not only is data outsourced to an untrusted environment, the access control policies are also outsourced to the proxy. With different re-encryption keys from the data owner, the proxy is able to present different views of outsourced data to different users according to their permissions without knowing the underlying plaintext.

With this scheme, the data owner is able to protect her data confidentiality in untrusted environment and control the access to her data with dynamic user groups. Besides, the data owner is able to get rid of heavy computation jobs for user access controls by offloading as much the *d*irty job of data re-encryption operations as possible to the proxy with small communication overhead.

However, a significant technique challenge here is that previous proxy re-encryption is not efficient enough to support large-scale data services in today's cloud computing. Although data encryption operation can be done offline at the data owner side, re-encryption operation by the proxy in the cloud and decryption operation at the client side, which are required by the proxy re-encryption algorithm, would affect user experience badly with its large computational overhead. Therefore, it is discouraged to directly apply this algorithm on the cloud.

In this work, we propose a dual cryptographic scheme, which inherent all the benefits from the previous proxy re-encryption algorithm while possessing efficient system performance. By integrating standard symmetric encryption and proxy re-encryption algorithm to pre-process and post-process the outsourced data, our scheme achieves efficient data encryption, re-encryption, and decryption operations. Being capable of efficiently re-encrypting the data, our scheme is able to support flexible access control and easy revocation on dynamic data user groups.

1.2.2 Secure Cloud-based Content Delivery Service with Proxy Re-encryption

A popular cloud service is cloud-based content delivery service. For cloud-based content delivery services, there are new security and system requirements. First, the content security should be realized by the content provider who uses public cloud services, instead of the cloud service provider [19, 111]. A content provider needs to encrypt her content with keys that are out of the reach of the cloud provider. Second, the access control policies should be flexible and distinguishable among users with different privileges to access the content. Each piece of content may be shared by different users or groups, and users may belong to multiple groups. Third, the number of redundant copies of the content cached in the content delivery network should be minimum in order to preserve efficiency of content distribution via the content delivery network. A user may earn benefits from the cache of encrypted content in the content delivery network of other users who have the same privilege. Multicast security [48] aims to address the confidentiality of content sharing with dynamic user groups. However, conventional multicast and broadcast involve only two types of entities: multicast/broadcast center and users. The content center belongs to the content provider or is fully trusted by the content provider. Their setting differs from our cloud-based model, which involves a semi-honest cloud provider to assist the content provider and the users. The earlier proxy-based encryption scheme for secure file systems [28] seems to work with semi-honest servers, but it fails to consider frequent key revocation problem, which is required by cloud-based data sharing systems. Therefore, we need a system with stronger content security

guarantees and more flexible user and key management mechanisms.

With above problems and design goals in mind, we propose CloudSeal, an end-to-end solution for secure content storage and delivery via the public cloud. By end-to-end, we mean that the content is encrypted at cloud-based storage and delivery channels. Only authorized end users or the content provider can decrypt it. CloudSeal ensures content confidentiality and content forward and backward security. CloudSeal utilize our proxy-based cryptographic scheme to protect content confidentiality and efficiently control the access to the outsourced content. Then, CloudSeal seamlessly integrates k-out-of-n secret sharing mechanisms to guarantee content forward and backward security.

Besides these security properties, CloudSeal designs a novel content delivery model which further enables efficient content distribution and flexible content access control mechanisms. CloudSeal splits the ciphertext of the content stored in the cloud into two parts, so that the proxy only re-encrypts a very small part of ciphertext, and the large portion remains unchanged. The proxy guarantees two important properties of the content: 1) the content to be downloaded by the subscribers is always encrypted with the latest re-encryption key; 2) there is only *one* copy of the content stored in the content delivery network. These features enable the efficient cache mechanism during content distribution and achieve fast content distribution. For flexible content access control mechanisms, CloudSeal separates the distribution of the subscribers' decryption key from that of the content to enforce flexible authorization policies. Only authorized users can obtain the latest decryption key, and the content provider maintains the control of issuing new keys. We design k-out-of-n secret sharing and broadcast revocation protocols to renew the shared secret key in

a scalable fashion.

1.2.3 Secure Cloud-based Data Processing Service with Vulnerable Cloud Virtualization Environment

Another typical cloud service is cloud-based data processing service. In this service, the data owner rents computing resources from cloud service providers, such as storage units and virtual machines, then stores and processes her data in the cloud. However, semi-honest cloud service providers and malicious cloud customers have become the big obstacles for it. The cloud providers may provide satisfied services to their customers while they may also attempt to peak the stored or processed data once the data is exported from the data owner to the cloud; the malicious cloud customers might steal cryptographic keys used by their neighbor virtual machines with recently discovered malicious side-channels. These threats bring extra security requirement in cloud-based data processing service in a way that not only does a data owner needs to protect confidentiality and support access control to the outsourced data, the data owner also needs to protect the cryptographic key used in an insecure cloud infrastructure. Unfortunately, current security solutions for cloud-based data processing services fail to support secure key usage in current cloud environment.

To fight against cross-VM side channel attack launched by malicious cloud customers in the public cloud virtualization environment, we observe that the stealing behavior involving in such attack can only be accomplished if the secrecy, such as data decryption keys for AES algorithm, is loaded into the victim machine's physical component, e.g. CPU cache. With this observation, a simple

solution to conquer such attack is to use one-time secret keys in the victim machine: each secret key is discarded once it is been used. However, this approach brings two major challenges: 1) complicated key management due to the large amount of short-lived data decryption keys; 2) heavy computation requirements over encrypted outsourced data by the data owner with frequent updating data decryption keys. Without overcoming these two challenges, it is impractical to use one-time secret key in the current public cloud environment.

In this work, we introduce *CloudSafe*, a novel approach that offers security means to protect outsourced data from both semi-honest cloud providers and malicious cloud customers. To efficiently support one-time secret key strategy, CloudSafe utilizes the proposed cryptographic scheme with acceptable cryptographic computation overhead and storage cost. To securely store and distribute one-time data decryption keys in the public cloud, CloudSafe design and develop a novel centralized key distribution framework. Seamlessly integrating cryptographic scheme and key management framework makes CloudSafe secure and practical to be used in today's untrusted public cloud environment.

Specifically, to distribute the one-time data decryption key from the data owner to the computing entities in the cloud, CloudSafe splits the process into two phases. The first phase is to distribute the key into the the framework in the public cloud data center where user's virtual machines are running over public network; the second phase is to distribute the key from the the framework to user's virtual machines through the intranet of the data center of a cloud. During the distribution, the key management framework guarantees that 1) the data decryption key is securely stored for the data owner; 2) the rented virtual machines are able to access the keys from anywhere in the

cloud data center. These features enable CloudSafe to support secure data decryption key usage in a dynamic and vulnerable virtualization environment in the public cloud.

1.3 RoadMap

The remaining part of this dissertation is organized as follows.

Chapter 2 reviews previous researches that are close to our research in section 2.1 and preliminary techniques we use in this dissertation in section 2.2. Specifically, we divide previous research into four topic in section 2.1. We presents the state of the art security research in cloud computing area in section 2.1.1. We go through outsourced data security problems and solutions in section 2.1.2. We look into security about content delivery in section 2.2.3. At last, we visit the traditional side-channel attack and its countermeasures in section 2.2.4. For preliminary techniques, we overviews proxy re-encryption algorithms in section 2.2.1 and then describe the concepts of auxiliary techniques in section 2.2.2 and complexity assumption that we will use in the following chapters in section 2.2.3.

Chapter 3 introduces our proxy-based data outsourcing scheme. In section 3.1, we depict the security problem we will solve with the new scheme and present the motivation of this work. In section 3.2, we demonstrate details of the scheme. Section 3.2.1 defines the main algorithm; section 3.2.2 presents the construction of the algorithm; section 3.2.3 demonstrate its usage protocols in the cloud; section 3.2.4 analysis the security of the scheme. We present our implementation and evaluation in section 3.3 and summarize this chapter in section 3.4.

Chapter 4 proposes *CloudSeal*, a secure data sharing scheme for cloud-based secure content delivery service. In section 4.1, we describe the problem we aim to solve and motivation of this work. In section 4.2, we present the scenario and design goals of CloudSeal. In section 4.3, we elaborate details of CloudSeal including the overview in section 4.3.1, the user management protocols in section 4.3.2, and security analysis in section 4.3.3. In section 4.4, we demonstrate the implementation and evaluation of CloudSeal. We then conclude this chapter in section 4.5.

Chapter 5 introduces *CloudSafe*, a framework for secure data processing in vulnerable cloud virtualization environment. We present the problem and motivation of this work in section 5.1. In section 5.2, we describe the setting and design goals of CloudSafe. Section 5.3 overviews the path of CloudSafe design. Section 5.4 provides the cryptographic scheme of CloudSeal with definitions in section 5.4.1 and construction discussion in section 5.4.2. We presents the key management component of CloudSafe in section 5.5 with overview in section 5.5.1, key distribution strategy in section 5.5.2, usage with virtual server lifecycle in section 5.5.3, and discussion in section 5.5.4. In section 5.6, we demonstrate the implementation and evaluation results of CloudSafe. Section 5.7 presents chapter summary at last.

Chapter 6 concludes this dissertation and presents the future work based on this dissertation.

Chapter 2

Literature Review and Preliminary Techniques

2.1 Literature Review

This dissertation focuses on security problems in cloud computing. Specifically, it involves outsourced data security in cloud storage, secure and efficient cloud-based content delivery service, and secure cryptographic key usage in cloud-based data computing service. In this chapter, we review the previous state of the art research in the related areas in four individual sections.

2.1.1 Security in Cloud Computing

Along with its rapid growth, cloud computing becomes a new hotbed for malicious and illegal activities. Not only do old attack tricks can successfully compromise cloud systems, but also, newly security problems targeting cloud system specifically, are emerging. However, due to the complexity of cloud systems, traditional solutions might not be a good fit to solve the old ones let alone the newly discovered ones in the cloud. In this section, we review modern researches on security problems in the area of cloud computing with two categorize in the following sections.

Storage Security

Many cloud security researches are data-centric security researches, which focus on confidentiality, integrity, and access control of outsourced data when they are stored in the cloud storage. To protect outsourced data confidentiality in the cloud storage, Kamara et cl. in [77] discussed a general idea for securing cloud storage by cryptographic techniques. They examined several recent advanced cryptographic techniques that were promising to realize this idea. One year later, they designed and implemented CS2 [112], a cryptographic cloud storage system with support of efficient standard and assist keyword search on the ciphertext. The search utility over encrypted cloud storage are also mentioned in [50, 86, 129], Wang et cl [129] constructed an order-preserving symmetric encryption algorithm to support ranked keyword search over encrypted data in the cloud. In 2011, Cao et cl. [50] designed an extensive solution that is able to preserve search privacy and support multi-keyword ranked search on encrypted data in the cloud. Li et cl. [86] addressed the problem

of authorized private keyword searches on the encrypted personal health data in the cloud. Zhou et al [152] focused on secure data sharing and processing in the distributed environment, including secure query processing, secure data sharing, and forensics on data operations in the distributed system. To support integrity check on the outsourced data storing in a third party cloud storage, Popa et cl. [106] built CloudProof, a proof-based cloud storage which enabled the customer to verify integrity , write-serializability, and freshness of her data after she stores them in the cloud. Wang et al. in [130, 131, 132] addressed the desire of auditing data integrity when it was stored in third party's cloud storage. They designed a public verification model by improving existing Proof of Retrievability model. This model was able to not only allow a third party auditor to perform intensive auditing operations but also support auditing services with dynamic data. They also demonstrated the construction of a privacy-preserving third party auditor to check the integrity of outsourced data with public cloud storage services. Benson et cl. [35] proposed an approach that enables the cloud users to verify data replicate services provided by the cloud provider. Stefanov et al.[121] built Iris, a scalable system to provide integrity guarantee for large outsourced file system in the cloud. To offer access control over outsourced data in the cloud storage, Yu et cl [146, 147] proposed specific solutions with help of attribute-based encryption and proxy re-encryption to secure and control the access to the outsourced data in the cloud storage. They used either KP-ABE or CP-ABE algorithms to associate sets of attributes to the ciphertext stored in the cloud storage and secret keys kept by users. Only the secret key which contained more attributes than the required attributes by the ciphertext was able to decrypt the ciphertext. Also they employed a semi-honest proxy in the cloud to reduce the workload at the data provider side. Our work focused

on confidentiality and access control over outsourced data in the public cloud. The uniqueness of our solutions is that we combine both cryptographic scheme and systematic ways to develop efficient and practical solutions for specific cloud services.

Recently, Zhang et cl. [148] proposed a solution that could take advantage of plentiful computational resources from cloud computing while be aware of data privacy through hybrid cloud infrastructure. With the rapid development of biological technologies, we believe more and more computational operations will be outsourced to the cloud so as to take advantage of its abundant computational resources. We will focus more on this area in our future work.

Infrastructure Security

Infrastructure-as-a-Service is one of the important services emerging in the era of cloud computing. However, it also becomes a hotbed for malicious attackers. Many new threats and attacks are found recently in cloud IaaS services due to its special infrastructure characteristics. Ristenpart et cl. in [109] revealed the possibility of stealing information in a third-party compute clouds. In their research, they found out several rules that Amazon was using to map their compute service EC2 instances, such as same IP address ranges was likely from the same available zones. These rules led them to successfully locates an instance in the cloud infrastructure, co-resident two instances on the same physical machine in the cloud, and finally steal information from cross-VM exploits with co-residence. Later in 2013, Zhang et cl. [150] elaborated details of a real attack on constructing side-channel and stealing ElGamal data decryption key in the cloud environment. Besides vulnerabilities in the lower infrastructure of cloud computing, researchers [31, 46, 119] have

found privacy risk existing in Amazon machine images (AMI). In [46], an automated tool using only public interfaces from the cloud provider can extract sensitive informations from the AMIs listed in the public cloud app store. In addition, authors in paper [31] pointed out that out-of-date softwares installed in pre-configured AMIs and remaining history information in a used AMIs such as browser history were also the reasons that made AMI vulnerable to the attackers. Due to these vulnerabilities and exploits, virtualization environment in cloud computing is experiencing more dangerous situation than ever before when virtualization environment was secure. Also without a secure virtualization environment, traditional security approaches for applications or data that rely on secure running environment are under the risk of uselessness. Moreover, researchers in [119] demonstrates that compromising the control interfaces of a cloud infrastructure services is possible in real public cloud environment. Reconsideration on the relationship between the infrastructure security and application/data security is needed in designing effective security mechanisms for cloud computing. In this work, we cast our initial thoughts on this issue into a solution that aims to provide strong security guarantees towards these threats.

In response to those threats, researchers from both industry and academic have proposed several solutions to secure current cloud platform [7, 120]. Researchers in IBM corporation have proposed and realized virtual trusted platform module (vTPM) for Xen virtualization environment to explore the usage of trusted platform module (TPM) in the cloud environment. vTPM provides additional modules in both hypervisor and virtual machines to enable the functionality of TPM been applicable in the dynamic and distributed cloud environment. Besides TPM, Song et cl. proposed a general framework to build data-protection-as-service(DPaaS) in the cloud. DPaaS inte-

grates different protection techniques to provide a combined multi-tier protection mechanisms for the current cloud. CloudSafe is more specific with main focus on supporting data confidentiality against unauthorized users in cloud storage and secure key usage within a vulnerable virtualization environment. Our work of securing key usage in cloud-based data processing service focuses on the recent discovered cross-VM side-channel attack [109, 150] and proposes a generic framework, CloudSafe, to support secure and efficient key storage and usage in the vulnerable cloud virtualization environment. CloudSafe is compatible with vTPM in which CloudSafe is able to seal its short-lived data decryption key. Besides, CloudSafe is more effective in avoiding stealing from malicious cloud customers through hidden side-channel by its one-time key strategy. To extend CloudSafe for more security protection mechanism, DPaaS is a guideline for us in the future work.

Besides newly developed attack in cloud computing, traditional security issues existing in the hypervisor are still big threats against cloud computing, especially those practical threats [16] existing in Xen hypervisor since it is one of the most popular hypervisors used in current cloud infrastructure. Due to its advanced privileges, attacks against hypervisor may exert more serious threats to the cloud computing platform than that against virtual machines. Previous researches have already addressed these issues [54, 84, 93, 123] by disaggregating the administrative domains of the hypervisor. Most recently, Butt et al. [47] built a self-service cloud computing model by splitting administrative domains in order to protect security and privacy of virtual machines. Our current work assumes that the hypervisor is fully trusted. With untrusted hypervisor in reality, our schemes needs further research for practical use.

2.1.2 Outsourced Data Security

Before the emergence of cloud computing, researchers have studied outsourcing data security for a long time. Many researches on outsourcing data security are about data storage security and access control [27, 55, 56, 57, 58, 89, 133]. Related to our prior research, authors in [57, 58] proposed the idea of over encryption algorithms with which two layer encryption were applied to outsourced data to achieve confidentiality and access control. Usually, the inner layer of encryption was to protect data confidentiality while the outer layer encryption was for access control purpose. Our algorithm made this idea come true with two concrete encryption algorithms specially for cloud computing with small overhead and easy scalability. Moreover, other researches such as data service security and encrypted data utility [59, 137, 141] have also been studied by many researchers. For example, paper [59] focused on cryptographic-based database outsourcing; paper [137] was about auditing utilities on encrypted data; paper [141] discussed about secure outsourcing aggregation services; researchers in [52, 78, 118, 145] were interested in search utilities on encrypted data.

More specifically, lots of efforts have been made on secure outsourcing data to untrusted storage or file systems. For example, Blaze [38] suggested to encrypt sensitive data before put them into local disk or remote server in 1993. He designed and implemented CFS as a Unix-like encrypting file system. Stein et al. [122] focused on data integrity in local storage. Mazières built SFS, a secure file system aimed to provide a single global file system without a separated key management for each individuals. Later on, Mazières et cl. [85, 91] designed and realized SUNDR

with focus on data integrity and freshness with untrusted servers. FARSITE in [21] used symmetric encryption algorithm and hash techniques to enable secrecy and integrity of data stored in an incompletely trusted environment. Sirius [74] and Plutus [76] were two secure file system with untrusted remote servers which use modern cryptographic techniques and customized key management mechanism to protect data security and guarantee data integrity with untrusted environment. To secure distributed file system, hash function were used to guarantee data integrity [51, 92], or to build Merkle tree [21, 62, 76, 97, 104] for efficient security mechanism.

In addition to outsourced data security, many researches have focused on outsourcing data computation to untrusted third parties. Dating back decades when the end devices were lack of computational resources, the need of securely outsourcing computation operations to untrusted auxiliary devices attracted many cryptographic experts in devising secure and efficient solutions to meet this need [24, 25, 26, 33, 34, 79, 87]. Although traditional solutions to outsourced data security problems might not fit for that in cloud computing, reviews of the trace of their development is helpful for finding solutions to current problems. Atallah et cl. [26] proposed several data disguise techniques to support secure outsourcing of scientific computations. Their solutions [26] required data preprossesing before data was outsourced and result postprocessing after the results are sent back from the remote servers. Not only scientific computations, Atallah et cl. [25] designed a protocol to outsource computational operations in a privacy-preserving manner in 2005. They focused on solving large-scale problem such as sequence comparison on grid computing with which the end device could avail itself with the abundance resources through elsewhere on the network.

These researches have stressed many significant data security problems with untrusted remote

environment. However, data security problem in the era of cloud computing is more serious and complicated than ever before in its size, scale and utility. Traditional solutions to outsourced data security may not fit in cloud paradigm. We need to reconsider these security problems under a new epic and seek more suitable solutions for current situation.

2.1.3 Security in Content Delivery

With the emphasis of Internet security, many security-embedded network protocols [8, 13, 14] are designed to support secure content delivery over public network. However, content delivery over content delivery network introduces additional security problems that are beyond the duty of traditional network security protocols. Secure multicast communication [48, 153] and conditional access systems [103, 125] address similar security problems as ours in distributing content to dynamic user groups and key management. For example, researchers in paper [53] also utilized proxy re-encryption algorithm and proposed proxy re-encryption based secure multicast mechanisms to achieve scalability and containment. Several proxies (routers), usually during the content transmission, transitively converted ciphertext data with re-encryption keys assigned when building a multicast network. When a user was to be revoked or a proxy (router) was off the network, corresponding proxy re-encryption keys and group secret keys need to be updated. The problem solved by our solution is different from them due to the cache properties in content delivery network, which requires more efficient and flexible secure content delivery and user management mechanisms. Besides proxy re-encryption, other researches on secure multicast communication mainly focus on sourced authentication [22, 40, 100, 102, 110, 136] and group key management

issues [44, 45, 67, 68, 69, 70, 135]. For source authentication, Rohatgi et al. [110] presented a hybrid signature scheme to fast authenticate packet source for multicast; researchers in [102] designed TESLA to improve authentication efficiency; Boneh et al. [40] cryptographically prove that it is difficult to build short and efficient code to check message integrity in multicast environment. For group key management, Rafaeli et al. [107] categorized group key management into three classes according to their structure; Naor et al. [94] focused on key revocation problems with stateless receiver; Sherman et al. [66, 116] worked on key establishment problem. In our research, we investigated the extra security and system issues arose from cache property in content delivery network. Traditional solutions to key management for multicast and conditional access system may be suitable for content delivery network. However, security and system issues brought by cache property in the content delivery network need newly security mechanisms which is focused in our research.

2.1.4 Side-channel Attack and Its Countermeasures

Side-channel attacks on cryptosystem have been mentioned in many previous works. Koeune et al. [81] described side-channel attack using timing informations on AES with vulnerable implementation. Kelsey et al. [80] pointed out that cache hit ratio could be used to attack large S-box cipher. Page theoretically discussed the side-channel attacks by using cache misses in [98] and their countermeasures on smartcard [99]. Inspired by this attack, Bertoni et al. [37] and Aciicmez et al. [20] realized similar attack by using power analysis on a MIPS microprocessor. Also in 2002 and afterwards, Tsunoo et al. [127, 128] presented attacks due to time effect of cipher memory

lookups collisions. Countermeasures against such attack were proposed in [23, 83]. Bernstein et al. [36] realized side-channel attack by exploiting timing variability of cache effect in AES. Percival et al. [101] realized side-channel attack on RSA. Most recently, Tromer et al. [126] focused on side-channel attacks on AES due to the inter-process information leakage. However, due to the pervasiveness and stealthy characteristics of side-channels, directly blocking all the side-channels existing in a system is difficult, especially in a system as complicated as the public cloud. Recently, Dr. Ford seeks to solve this problem by deterministic execution and pacing queues to eliminate the sharing and concurrency in the system [29, 60, 61, 65]. With leakage free execution in operating system, it was hard to build covert channels in such system. This solutions is promising but still at its preliminary stage. Besides bad effect of side-channels, researchers in [149] designed a system to utilize side-channels as a defensive tools to exclude malicious cloud customers who ran a co-resident virtual machines as the victim one. In our research, we proposed CloudSafe which aims to provide means to bypass such attack with the one-time secret key strategy while preserving the system efficiency with acceptable overhead in the cloud. Although new side-channels might be built in future, CloudSafe is secure against such attack to prevent them from stealing cryptographic keys in the current cloud environment.

2.2 Preliminary Techniques

In this section, we present several techniques that we have used in our works, including concepts about proxy re-encryption, secret sharing, bilinear map, and complexity assumption.

2.2.1 Proxy Re-encryption

Definition

Generally, a proxy re-encryption algorithm transforms ciphertext c_{k1} to ciphertext c_{k2} with a key $rk_{k1\to k2}$ without revealing the corresponding cleartext, where c_{k1} and c_{k2} can only be decrypted by different key $k1$ and $k2$, respectively. $rk_{k1\to k2}$ is a re-key issued by another party, e.g., the originator of ciphertext c_{k1}. In proxy re-encryption, a proxy is set up to perform *Re_encryption* operation; other operations are executed at two end users side. The proxy is usually semi-trusted. It does not necessarily belong to any end users. It conforms to the assigned operations but it may try to peak the underlying plaintext during the execution. With proxy re-encryption, the efficiency of traditional decrypt-and-then-encrypt procedure can be significantly improved.

A formal definition of typical proxy re-encryption scheme is presented with the following five algorithms.

- *KeyGen.* This algorithm generates a pair of key $\{PK_A, SK_A\}$. PK_A is A's public key and SK_A is A's private key;
- *Encryption.* This algorithm takes PK_A and message m as inputs and outputs ciphertext C_A;
- *ReKey.* This algorithm takes PK_A and PK_B as inputs and generates a rekey $rk_{A\to B}$;
- *Re-encryption.* This algorithm takes $rk_{A\to B}$ and ciphertext C_A as inputs and outputs ciphertext C_B.

- *Decryption.* This algorithm takes SK and C as inputs and outputs message m.

Overview

After been proposed by Blaze et.cl. in 1998 [17], the notion of *proxy cryptography* has been largely developed. In [17], a bidirectional and transitive proxy cryptography based on El Gamal scheme was built with the delegation key *b/a mod q*. The proxy used this delegation key to convert ciphertexts from Alice to Bob and vice versa. Besides, with two correlated delegation keys, for example *b/a* and c/b, the proxy was able to automatically divert ciphertexts from Alice to Carol. The proxy had to be fully trusted in not colluding with any participants with secret keys. Jackson [73] developed an asymmetric proxy re-encryption scheme based on El Gamal model with consideration of dishonesty of the proxy. According to him, the delegation key should be divided into many shares and each proxies should only control one share. Given this approach, the delegation key was secure as long as some proxies are honest in the system. Zhou [151] presented a similar research on this topic in 2005. Ivan and Dodis [72] built both bidirectional and unidirectional proxy re-encryption schemes with El Gamal, RSA, and an IBE scheme. They solved the problem of transitivity in proxy re-encryption, however they still suffered from collusion problem between the proxy and the delegatees. Ateniese et.cl. [28] solved the collusion problem by building proxy re-encryption with a bilinear map settings. Their algorithm inherent unidirectional and complexity properties of bilinear map such that only a weak version of Alice's secret key would be recovered under the collusion between the proxy and Bob. More and more proxy re-encryption schemes were proposed recent years[49, 64, 90, 114, 115, 143] In this manuscript, we are interested in improving proxy

re-encryption techniques to be practical enough so that the data owner can use them to protect outsourced data security in the cloud. We will elaborate our work on this topic in the next chapter.

2.2.2 Auxiliary Techniques

The other techniques that we use in the following work are presented as follows.

Bilinear Map [41, 43]: Let $\mathbb{G}$, $\mathbb{G}_T$ be two multiplicative cyclic groups of prime order *r*, we say $\hat{e}$ is an admissible bilinear map if: (1) computative actions in $\mathbb{G}$ and $\mathbb{G}_T$ are efficient; (2) for all $\alpha, \beta \in \mathbb{Z}_r$ of prime order r, $\hat{e}(g^\alpha, g^\beta) = \hat{e}(g, g)^{\alpha\beta}$; (3) $\hat{e}$ is non-degenerate, which means that not all pairs in $\mathbb{G} \times \mathbb{G}$ are sent to the identity in $\mathbb{G}_T$ by $\hat{e}$.

Secret Sharing [95, 113]: A k-out-of-n threshold secret sharing scheme is that a secret $S \in \mathbb{Z}_r$ shared by n users can be recovered, if the number of the secret shares exceeds the threshold k. The scheme utilizes a random polynomial P of degree $k-1$, where $P(x) \in \mathbb{Z}_r$ and $P(0) = S$. Given any k shares $< x_0, P(x_0) >, ..., < x_{k-1}, P(x_{k-1}) >$, one can use Lagrange interpolation formulas as follows to recover $P(0)$:

$$P(0) = \sum_{i=0}^{k-1} \lambda_i P(x_i), \text{ where } \lambda_i = \prod_{j \neq i} \frac{x_j}{x_j - x_i} \tag{2.1}$$

2.2.3 Complexity Assumption

The complexity assumption we use in this dissertation is described as follows.

Decisional Bilinear Diffie-Hellman (DBDH) Assumption Let $\mathbb{G}$ and $\mathbb{G}_T$ be two bilinear group

with an efficient computable pairing $\hat{e} : \mathbb{G} \times \mathbb{G} \leftarrow \mathbb{G}_T$, let g be a random generator of $\mathbb{G}$, and let $a, b, c \in \mathbb{Z}_p$. The DBDH assumption is that, for any probabilistic polynomial-time algorithms Γ, there is non-negligible advantage to decide a tuple of values $\left(g, g^a, g^b, g^c, T\right) \in \mathbb{G}^4 \times \mathbb{G}_T$ whether $T = \hat{e}\left(g, g\right)^{abc}$ or T is randomly chosen from $\mathbb{G}_T$. This assumption is also used by several identity-based encryption schemes [39, 42, 134]

Chapter 3

Proxy-based Secure Data Outsourcing with Public Cloud

In our service-centric security solutions, cryptographic techniques serve as the basis of outsourced data protection in the cloud. In this chapter, we present a novel proxy-based secure data outsourcing scheme and demonstrate its usage in the cloud.

3.1 Problem Description and Motivation

Secure outsourcing data to an untrusted server has been studied for decades. Researchers have proposed many solutions to protect confidentiality and control the access to the outsourced data. Among them, proxy-based re-encryption scheme designed by Ateniese [28] is an advanced model

known for its achieved properties. In this model, a proxy is usually sitting in the untrusted environment between a data provider and users. Not only is data outsourced to an untrusted environment, the access control policies are also outsourced to the proxy. With different re-encryption keys, the proxy is able to present different views of outsourced data to different users according to their permissions without knowing the underlying plaintext. The scheme advances in three aspects. Firstly, the data owner is able to protect her data security in untrusted environment with dynamic user groups; secondly, controlling users access to her data after she has outsourced it to the cloud is simple and flexible; thirdly, the data owner is able to get rid of heavy computation jobs for user access controls by offloading as much the *d*irty job of data re-encryption operations as possible to the cloud with small communication overhead.

However, a significant technique challenge here is that previous proxy re-encryption is not efficient enough to support large-scale data services in today's cloud computing.Although data encryption can be done offline at the data owner side, re-encryption operation by the proxy at the third-party and decryption operation at the client side, which are required by the proxy re-encryption algorithm, would affect user experience badly with its large overhead. Therefore, it is discouraged to directly apply this algorithm on the cloud. In this work, we propose a dual cryptographic scheme, which inherent all the benefits from the previous proxy re-encryption algorithm without sacrificing much user experience. To devise our scheme, we first split the large overhead from previous algorithm into each pairing operations that are executed during encryption, re-encryption, and decryption as shown in table 3.1. The table tells us that it takes considerable time to generate random number from $\mathbb{G}$. Unfortunately, this operation is repeatedly executed for all the block of a file,

which delays the whole execution time of the algorithm in encryption and re-encryption operations. With this observation in mind, we then carefully reduce this operations into once in our algorithm by adding one layer of block cipher operations before proxy re-encryption operations to protect data confidentiality so as to improve its efficiency as well as preserving the security. With this extra layer of encryption algorithm, our scheme is able to achieve both security and efficiency while preserving the functionality of proxy re-encryption algorithms. In the remaining part of this chapter, we describe details of our two-layer schemes. Although we implement a certain block cipher and proxy re-encryption algorithm for our scheme, these two layers of cryptographic methods are not necessarily sticking to specific proxy re-encryption algorithm or block ciphers. Other combinations can also be used to symbolized our dual cryptographic scheme as long as the inner layer is able to protect data confidentiality and the outer layer can provide access control to outsourced data.

3.2 The Scheme

3.2.1 Algorithm Definitions

Let $\mathbb{G}$, $\mathbb{G}_T$ be two multiplicative cyclic groups of prime order *r*, and $\hat{e}$ be a bilinear map satisfying the three properties: (1) $\forall g \in G$ and $\forall \alpha, \beta \in \mathbb{Z}_r$, $\hat{e}(g^\alpha, g^\beta) = \hat{e}(g, g)^{\alpha\beta}$ (2) $\hat{e}(g, g) \neq 1$ (3) $\hat{e}$ can be computed efficiently.

Let π be a secure block cipher including three algorithms: (1) $\pi.KeyGen\left(1^\theta\right)$ takes a random pa-

Operations	Time (ms)
Random Number Generation in $\mathbb{Z}_r$	0.598
Random Number Generation in $\mathbb{G}$	5.734
Multiplication in $\mathbb{Z}_r$	0.0000209
Pairing in $\mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$	6.374
Exponentiation between $\mathbb{G}$ and $\mathbb{Z}_r$	4.972
Exponentiation between $\mathbb{G}_T$ and $\mathbb{Z}_r$	0.719
Inversion in $\mathbb{Z}_r$	0.000411
Division in $\mathbb{G}_T$	0.00271

Table 3.1: Executing time of operations in proxy re-encryption in [28]. The algorithm is implemented with `pbc` library [12] with `TYPE E` elliptic curve on a desktop with Intel Duo CPU 2.93GHz 4GB RAM CentOS 2.6.

rameter θ as input and generates data encryption key DEK for encryption; (2) $\pi.SEnc\left(DEK, M\right)$ takes data encryption key DEK and original message $M \in \{0,1\}^*$ and outputs a ciphertex C_s; (3) $\pi.Dec(key, C)$ decrypts the ciphertext C by key key and outputs the original message M.

Let Pro be a proxy re-encryption cipher including five algorithms: (1) $Pro.Setup\left(1^{\lambda}\right)$ takes security parameter λ as input and returns system parameters $params$, the master secret key $MK \in \mathbb{Z}_r$; (2) $Pro.KeyGen\left(1^{\varsigma}\right)$ takes random parameter ς as input and outputs the decryption key $AK \in \mathbb{Z}_r$; (3) $Pro.PEnc\left(params, MK, DEK, M\right)$ takes $params$, MK, and M as inputs and returns a ciphertext C_p; (4) $Pro.ReKey\left(params, MK, AK\right)$ takes $params$, MK, and AK as inputs and returns a re-encryption key $rk \in \mathbb{G}$; (5) $Pro.ReEnc\left(params, rk, C_p\right)$ takes $params$, re-encryption key rk and ciphertext C_p as inputs and returns C_r; (5) $Pro.Dec\left(params, key, C\right)$ takes $params$, a private key $key \in \{MK, AK\}$, and $C \in \{C_p, C_r\}$ as inputs and outputs $\{C_s, DEK\}$.

Correctness. These algorithms must satisfy the following correctness requirements.

1. $\pi.Dec\left(key, \pi.SEnc\left(DEK, M\right)\right) = M$ if $key = DEK$.
2. $Pro.Dec\left(params, key, Pro.PEnc\left(params, MK, DEK, M\right)\right) = \{C_s, DEK\}$ if $key = MK$.
3. $Pro.Dec\left(params, key, Pro.ReEnc\left(params, rk, C_r\right)\right) = \{C_s, DEK\}$ if $key = AK$ and $rk = Pro.ReKey\left(params, MK, AK\right)$.

3.2.2 Main Construction

The detailed construction of our proxy-based data outsourcing scheme is presented as follows.

Pro.Setup(1^λ)**:** The data owner initializes the system parameters:

1. Given the secret parameter λ, generate cyclic groups $\mathbb{G}$ and $\mathbb{G}_T$ of prime order r, let g be a generator randomly selected from $\mathbb{G}$, and $\hat{e}$ be a a bilinear map : $\mathbb{G} \times \mathbb{G} = \mathbb{G}_T$;

2. Generate random master secret key $MK \in \mathbb{Z}_r$ and keep it private;

3. Outputs $params = (\mathbb{G}, \mathbb{G}_T, \hat{e}, g)$. The message space $M \in \{0,1\}^n$.

When outsourcing a file to the cloud, the data owner first calls $\pi.KeyGen$ to obtain a data encryption key $DEK \in \mathbb{Z}_r$. Note that different files have different data encryption key DEK. Then she encryptes message M by calling $Pro.PEnc$ with master key MK and data encryption key DEK as follows:

Pro.PEnc(params, MK, DEK, M): The data owner encrypts the given plaintext M and DEK as follows:

1. Randomly select a parameter $h \in \mathbb{Z}_r$ and calculate $Z^h = \hat{e}(g,g)^h$ where $Z = \hat{e}(g,g)$;

2. Call $\pi.SEnc$ to encrypt message M with data encryption key DEK and outputs the first-level ciphertext C_s;

3. Generate the second-level ciphertext $C_p =< u, v, w >$ where $u = g^{MK \cdot h}$, $v = Z^h \cdot DEK$

and $w = (w_1, \cdots, w_N) = (Z^h C_{s1}, \cdots, Z^h C_{sN})$, where C_{s1} is a data block of C_s and the total number of data block is N.

Although the order of two level ciphertext generation is fix, the inner block cipher π, for example the advanced encryption stand (AES) used in our implementation and evaluation, that is used to generate the first-level ciphertext is up to data owners. Data encryption key DEK of each data is enclosed in the outsourced ciphertext for its own security and to eliminate runtime communication cost. Only one decryption key is needed to decrypt the ciphertext from the above dual encryption scheme.

When a user wants to access the data stored by the data owner in the cloud, the data has to be re-encrypted before it is sent to the user in order to ensure that only authorized data is accessed by the user. Re-encryption operation transforms the second-level ciphertext by the master secret key MK to a ciphertext that can be decrypted by the user's secrecy AK, which can be done by two steps. First, the data owner calls $Pro.ReKey$ to compute a re-encryption key rk according to the user's secrecy AK. Users can obtain her secrecy from third party authority or can be assigned by the data owner with $Pro.KeyGen$ function. Then, she invokes $Pro.ReEnc$ with ciphertext C_p, which she stores in the cloud before, and outputs re-encrypted data C_r to the user. As it is time-consuming to download C_p from the cloud before re-encryption, usually a proxy is set up by the data owner in the cloud to perform the second step. Details of $Pro.ReKey$ and $Pro.ReEnc$ are presented as follows.

Pro.ReKey(params, MK, AK): The data owner computes the re-encryption key $rk_{MK \to AK} =$

$g^{AK/MK}$ with given MK and AK.

Pro.ReEnc(params, rk, C_p): The proxy in the cloud

1. Given $C_p = (u, v, w)$, compute $u' = \hat{e}(rk, u) = \hat{e}(g, g)^{AK \cdot h} = Z^{AK \cdot h}$ with $rk = g^{AK/SK}$ and $u = g^{SK \cdot h}$;

2. Output the ciphertext be $C_r =< u', v, w >$ where v and w are the same as that of C_p.

With re-encryption, users can not decrypt the stored ciphertext C_p in the cloud with her secrecy AK but decrypt the re-encrypted content C_r. Therefore, the data that is stored in the cloud is secure. Only the data owner can decrypt all of them with master secret MK.

To decrypt a ciphertext which is either a dual encrypted data C_p or a re-encrypted data C_r, it can be done as follows.

Pro.Dec(params, key, C): The decryption can be performed by the data owner or the user:

1. The data owner decrypts the data as below where the decryption key is MK:

   - Compute $Z^h = \hat{e}\left(u, g^{1/MK}\right)$ where Z denots $\hat{e}\left(g, g\right)$ and $C = C_p = (u, v, w)$;
   - Compute $DEK = v/Z^h$ and $C_s = w/Z^h$;
   - Call $\pi.Dec\left(DEK, C_s\right)$ to get the original message M.

2. The user decrypts the data as below where the decryption key is AK:

   - Compute $Z^h = u'^{1/AK}$ where $C = C_r = (u', v, w)$;

- Compute $DEK = v/Z^h$ and $C_s = w/Z^h$;
- Call $\pi.Dec\,(DEK, C_s)$ to get the original message M.

3.2.3 Usage Protocols

To integrate above cryptographic operations into the system, we design five protocols that are used to support communication between the involved three entities including the data owner, cloud storage, and users in the system. There are three actions in total: *Publish*, *Retrieve*, and *Delegate*. Each protocol between any two entities is defined as a tuple of $\langle sender, receiver, action\rangle$. *sender* is the entity who actively performs the action towards the receiver; *receiver* is the entity who passively receive the results by the action from the sender. For example, when the data owner wants to publish her data to the cloud storage, the data owner is the *sender* and the cloud storage is the *receiver* and the action is *publish*. We group the five protocols into two clubs: *Data* and *Control*, according to their purpose. Figure 3.1 shows processes of each actions between three entities. Details about each process in terms of protocol are described as follows.

Data

- `<owner, storage, publish>`: is a one time action for each data. To publish a data from the data owner to the cloud storage, the data owner calls $\pi.KeyGen$, $\pi.SEnc$, and $Pro.PEnc$ to generate data encryption key DEK, ciphertext C_s, and the outsourcable data C_p. Then she sends C_p to the cloud as step $P1 : C_p$ shown in the *a) Publish* part of the figure 3.1. This action only needs to be done once for each data at the data owner side.

Figure 3.1: Protocols of our scheme used in the cloud

- `<storage, proxy, retrieve>` is a simple one with only data delivery operations. The proxy retrieves the data C_p with the data name *fname* from the cloud storage as steps *R2.1* and *R3.1* shown in the figure 3.1. Only the proxy is able to access the data stored in the cloud.

- `<proxy, user, retrieve>` will invoke the `<storage, proxy, retrieve>` action and two controls during its process. To retrive a data from the proxy to a user, the application needs to initial a data request with the data name *fname* and its certified identifier id such as its public key. With *fname*, the proxy execute `<storage, proxy, retrieve>` action and retrieve encrypted data C_p from the cloud storage. Steps *R1:(id, fname) and R4:Cr* show this process in the figure 3.1. In the mean time, the proxy requests a re-encryption key rk from the data owner with id as steps *R2.2:id* and *C1:rk* shown in the figure 3.1. With C_p and rk in hand, the proxy calls $Pro.ReEnc$ to re-encrypted the data to produce C_r and sends C_r to the application.

Control

- `<owner, proxy, delegate>`: delegates the decryption ability to a cloud application from a data owner. The data owner calls $Pro.KeyGen$ function to randomly generates a decryption key AK, associates it with the application identifier id in a database, and then sends this decryption key AK to the application as step *D2:AK* shown in the figure 3.1.

- `<owner, proxy, delegate>` delegates the re-encryption operation to a proxy. Given an identifier id, the data owner first obtains the application decryption key AK by either invoking `<owner, proxy, delegate>` control or looking into the database, and then she calls $Pro.ReKey$ to yield the re-encryption key rk and sends rk to the proxy in the cloud.

3.2.4 Security Analysis

Our scheme performs two types of encryption algorithms to the data before outsourcing it. One is symmetric encryption algorithm to protect the confidentiality of the original content. The other is the proxy re-encryption algorithm proposed in the paper [28], which is executed on the resulting ciphertext from the symmetric encryption to enable flexible user access control of the ciphertext.

Content Confidentiality

As we make no change to the symmetric encryption algorithm before applying it to the original message, our scheme is as strong as the symmetric encryption algorithm in protecting confiden-

tiality of the original plaintext, given the data encryption key is secure. Therefore, our scheme achieves the confidentiality of the encrypted content exposed in the public cloud. Furthermore, in any system state, the cloud service provider or an attacker cannot decrypt the cipher content with only re-encryption keys $rk_{SK \to uk}$ or user decryption key uk.

Data Decryption Key Security

The data decryption key is used by the symmetric encryption algorithm on the original message. We distribute the data decryption key by multiplying it with Z^h and embedding it into the dual-layer ciphtertext. The data decryption key is secure as long as the possibility of guessing out the random secret h by the attacker is negligible.

Master Key Security

The master secret key from proxy re-encryption algorithm is used on the ciphertext from the symmetric encryption algorithm. Our scheme directly inherits the master key security from the proxy re-encryption algorithm in the paper [28]. This means colluding delegatees are not able to compute or infer the master key SK of the delegator with the decryption key uk and re-encryption key $rk_{SK \to uk}$ in long term.

Security Risk.

Our scheme leverages the dual encryption scheme and uses the same h for all blocks of a content to improve the performance of proxy re-encryption algorithm. However, there is trade off between the security and performance. Considering a worst case that the ciphertext C is $\{0\}^*$, the resulting ciphtertext C_p will appears to be the same value after applying *Pro.PEnc* operation since we use the same h for all the blocks of a content. Therefore, the attacker are able to learn any patterns existing in the ciphtertext C. As long as the symmetric encryption algorithm is randomize enough, the possibility to have patterns in ciphtertext C is negligible.

3.3 Implementation and Evaluation

3.3.1 Implementation

We implemented aforementioned cryptographic algorithms ($\pi.KekGen$, $\pi.SEnc$, $\pi.Dec$, $Pro.PEnc$ $Pro.ReKey$, $Pro.ReEnc$, and $Pro.Dec$) based on cryptographic functions from the OpenSSL library [11] and the pairing-based cryptographic library (PBC) [12] with independent native processes. We chose Advanced Encryption Standard (AES) with 16 bytes key as our symmetric encryption algorithm and implement the `CFB` mode of `AES` to randomize the ciphertext of each data block of a file. For pairing-based encryption, instead of mapping an arbitrary M to a certain field $\mathbb{G}_T$, we mapped the elements in $\mathbb{G}_T$ to byte array in C language and use `XOR` operation in our implementation to produce w and v. We also realized the encryption, re-encryption,

and decryption operations of the previous proxy re-encryption algorithms proposed in [28] by the PBC library for comparison purpose.

3.3.2 Evaluation

To evaluate the efficiency of our scheme, we ran two types of comparison experiments. All of the experiments were running on a desktop with Intel Duo CPU 2.93GHz 4GB RAM CentOS 2.6. We ran each experiment six times and recorded the average time the operation takes. Figure 3.2 shows the comparison results of time consuming to execute different operations on different size of files by AES algorithms and the previous proxy re-encryption algorithms. It is easy to find out that the previous proxy re-encryption algorithms is too slow to be practical. We also ran our scheme with the same set of files and compared it with AES algorithms in encryption and decryption operations. We can find that our scheme is comparable to AES algorithm in these two operations although it is a little slower than AES algorithms shown in figure 3.3 and figure 3.4, which is more practical than the previous ones.

3.4 Chapter Summary

Proxy-based data outsourcing scheme is a promising direction to support secure and efficient data outsourcing in the untrusted cloud environment. However, previous versions of proxy re-encryption are hard to be directly applied in the cloud due to their efficiency problem. In this work, we solved this problem by designing a new proxy-based cryptographic algorithm which in-

Figure 3.2: Efficiency comparison between proxy re-encryption scheme proposed in [28] and standard AES algorithm.

Figure 3.3: Efficiency comparison of encryption operation between our scheme and standard AES algorithm.

Figure 3.4: Efficiency comparison of decryption operation between our scheme and standard AES algorithm.

tegrated the advantages of both symmetric encryption and proxy re-encryption. This integration assured that our scheme was able to provide a secure and efficient data outsourcing scheme for today's cloud usage. The experimental evaluation demonstrated the practicality of our scheme.

Chapter 4

Secure Cloud-based Content Delivery Service with Proxy Re-encryption

In this chapter, we focus on cloud-based content delivery services and present, CloudSeal, an end-to-end solution for secure content storage and delivery via public cloud. Through this chapter, we aim to demonstrate the necessity of considering systematic requirements while developing efficient security solutions for cloud services.

4.1 Problem Description and Motivation

Recent advance of Internet and information technology has shown two significant trends. First, media content has become the main Internet traffic. As predicated by Cisco, video streaming

will consume approximately 90% of Internet traffic in 2015 [71]. Second, utilizing elastic cloud computing and storage resources has become the trend for enterprises and consumer-oriented commercial services. Large scale content processing, storage, and distribution via public cloud infrastructures become promising for quality-guaranteed and cost-efficient media streaming services. Despite the increasing usage of cloud in applications and services, security issues have been the top concerns for cloud computing [18, 88]. Among them, how to maintain the confidentiality and privacy of outsourced content in the public cloud remains a challenging task. The security requirement becomes more complex with flexible content processing and sharing among a large number of users through cloud-based applications and services.

Previous work has addressed such problems in conventional distributed environments [82, 144]. For large scale cloud-based content sharing and distribution services, there are new requirements beyond this. First, the content security should be realized by the content provider who uses public cloud services, instead of the cloud service provider [19, 111]. A content provider needs to encrypt her content with keys that are out of the reach of the cloud provider. Second, the access control policies should be flexible and distinguishable among users with different privileges to access the content. Each piece of content may be shared by different users or groups, and users may belong to multiple groups. Third, the number of redundant copies of the content cached in the content delivery network should be minimum in order to preserve efficiency of content distribution via the content delivery network. A user may earn benefits from the cache of encrypted content in the content delivery network of other users who have the same privilege. Multicast security [48] aims to address the confidentiality of content sharing with dynamic user groups. However, conventional

multicast and broadcast involve only two types of entities: multicast/broadcast center and users. The content center belongs to the content provider or is fully trusted by the content provider. Their setting differs from our cloud-based model, which involves a semi-honest cloud provider to assist the content provider and the users. The earlier proxy-based encryption scheme for secure file systems [28] seems to work with semi-honest servers, but it fails to consider frequent key revocation problem, which is required by cloud-based data sharing systems. Therefore, we need a system with stronger content security guarantees and more flexible user and key management mechanisms.

4.2 Models and Design Goals

In this section, we reviews the scenario, design goals, and the main idea of CloudSeal.

4.2.1 The Scenario

Figure 4.1 shows a three-layer architecture for a typical content storage and distribution system over the public cloud infrastructure: a centralized *storage service* provided by a *cloud provider*, a *content delivery network (or service)* to accelerate content distribution over public network, and end users with variant devices as *subscribers (content consumers or clients)*. A *cloud provider* provides two cloud-based services: storage and content delivery. A *content provider* utilizes these two services to store, share, and distribute her content to multiple subscribers. The content provider and subscribers can access content via a cloud-based *application service*, which reads and man-

Figure 4.1: Schematic drawing of data flow in cloud-based storage and delivery services.

ages the content stored in the storage service via cloud storage APIs. The application service is an application deployed in the cloud by the content provider or a third party. The content provider can use multiple cloud-based services from different cloud service providers to host her application service, content storage service, and content delivery service. For example, Netflix [10] uses Amazon EC2 and S3 for content processing and storage, and uses multiple content delivery services such as Limelight, Level 3, and Akamia. Besides, in this work, we only focus on delivering stable data by content delivery network with dynamic user group. Delivering dynamic data, such as news feed and stock feed, is not in the scoop of this work. The subscribers access the content in the cloud through broadcast channels.

4.2.2 Security and System Goals

We trusts the content provider. The cloud service provider is *honest but curious*; that is, it follows pre-defined the protocol and operations, but it may actively attempt to gain the knowledge of the content. The content delivery service is also semi-trusted: it also may attempt to sniff content distributed and cached in the network, but it honestly performs all the operations and satisfies the quality of services, e.g., as specified in a service level agreement between the content provider and the delivery service provider. In addition, the cloud infrastructure (hardware and software) may be exploited by attackers who aim to obtain content [109].

We aims to protect the content with large size and leverages public cloud for storage and content distribution. We assume that a subscriber does not store any cleartext and ciphertext of the content permanently in her local device. Instead, she downloads the content from the cloud and the content distribution network when she wants to access. We further assume that a subscriber does not re-disseminate decrypted content and her share of secrets to unauthorized parties.

We summarize the security objectives and system objectives that we want to achieve as follows.

- Our solution should ensure content confidentiality in the cloud storage and content delivery network even under the collusion between the cloud provider and the revoked subscribers.
- Our solution should support dynamic group-based user authorization, i.e., a user may choose to join or leave a group, or to be revoked from a group by the content provider at any time. Only authorized users are able to obtain the cleartext of protected content stored in cloud or cached in network at any system state.

- Our solution should support flexible security policies including forward and backward security.
  - For *forward security*, a user cannot access content published before she joins a group.
  - For *backward security*, a user cannot access content that is published after she leaves or is revoked from a group.

For forward and backward security, Our solution can be configured to support either or both. For example, a user may be allowed to access any movie that has been released before she subscribes, but cannot access any content after being revoked. For another example, a family content sharing application may allow a family friend to only access shared photos published during a certain period.

Beyond these security objectives, our solution aims to achieve the following system and network performance requirements.

- Our solution should preserve the efficiency of content delivery network. In particular, it is desirable for the network to store a single copy of encrypted content at each system state.
- Our solution should support light-weight end storage cost. Only a small amount of storage should be required at the content provider side and the subscriber side to sustain user and key management of CloudSeal.
- Our solution should not affect user experiences at device side. The overhead of security mechanisms should be acceptable.

With above problems and design goals in mind, we propose CloudSeal, an end-to-end solution for secure content storage and delivery via the public cloud. By end-to-end, we mean that the content is encrypted at cloud-based storage and delivery channels. Only authorized end users or the content provider can decrypt it. CloudSeal ensures content confidentiality and content forward and backward security. CloudSeal utilize proxy-based data outsourcing schemes to outsource data in the cloud in order to protect content confidentiality. Then, CloudSeal seamlessly integrates proxy re-encryption and k-out-of-n secret sharing mechanisms to guarantee content forward and backward security. A proxy is employed in the cloud to transform encrypted content stored in the cloud storage to the delivery network or directly to the subscribers. The content provider updates re-encryption keys for legitimate groups and enforces shared secret keys among authorized subscribers, with which the content retrieved from the cloud can be decrypted.

Besides these security properties, CloudSeal designs a novel content delivery model which further enables efficient content distribution and flexible content access control mechanisms. CloudSeal splits the ciphertext of the content stored in the cloud into two parts, so that the proxy only re-encrypts a very small part of ciphertext, and the large portion remains unchanged. The proxy guarantees two important properties of the content: 1) the content to be downloaded by the subscribers is always encrypted with the latest re-encryption key; 2) there is only *one* copy of the content stored in the content delivery network. These features enable the efficient cache mechanism during content distribution and achieve fast content distribution. For flexible content access control mechanisms, CloudSeal separates the distribution of the subscribers' decryption key from that of the content to enforce flexible authorization policies. Only authorized users can obtain the

latest decryption key, and the content provider maintains the control of issuing new keys.

For user management, we design k-out-of-n secret sharing and broadcast revocation protocols to renew the shared secret key in a scalable fashion. As the subscribers access the content through broadcast channels, broadcast encryption based revocation protocol is more suitable than traditional key management mechanism to address the extra cryptographic problems brought by broadcast channel delivery. This is because that the ability to revoke users through each broadcast transmission can not be achieved by traditional key management mechanisms. We refer the readers to the paper [95] for more details.

The security mechanism and supported access control policies of CloudSeal differ from what Netflix adopts [105] in several aspects. First, the goal of CloudSeal is to protect the security of outsourcing content for the content provider, rather than to prevent digital rights of the content on subscribers' device in Netflix, which uses Microsoft DRM on subscriber side to ensure end-to-end content security and digital rights management, e.g., to prevent further dissemination of protected content by an end user. Second, the access control of CloudSeal supports various groups of subscribers with different privileges for different shared content, while Netflix currently supports only one (unlimited) plan for their video streaming service. Third, CloudSeal encrypts the content when storing in public cloud and distributing via content delivery network. Netflix only encrypts the content at the edge of the content delivery network.

Although CloudSeal is not designed for dynamic data delivering with content delivery network, it is still more advanced to apply CloudSeal than traditional user management for such usage given the user groups are dynamic. This is because of the ability of CloudSeal to preserve the cache

efficiency of the content delivery network, which makes the cache reusable with dynamic user group.

4.3 CloudSeal Scheme Details

This section presents the overview, user management and security analysis of CloudSeal in details.

4.3.1 Overview

Figure 4.2 shows the architecture of CloudSeal with three types of players: *cloud provider*, *content provider*, and *subscriber*.

- *Cloud Provider* provides two public cloud services: *storage service* for content storing and *content delivery* for content distribution. It also provides virtual infrastructure to host application services, which can be used by the content provider to manipulate the content stored in the cloud, or by the content subscribers to retrieve the content.

- *Content Provider* provides content to groups of subscribers, as well as user management. It uses cloud-based service from the cloud provider to store and distribute content.

- *Subscriber* is able to access the content stored in the cloud if she successfully subscribes to the content provider. The subscriber can decrypt delivered content and consume it with local software.

Figure 4.2: CloudSeal overview.

The application service provides web interfaces for the content provider to publish and manipulate content in the cloud, as well as for subscribers to retrieve content from the cloud. The content can be directly accessed by consumers from the cloud storage service via storage APIs provided by the cloud service provider, or be cached in the delivery network once it has been accessed to save bandwidth and communication cost for the sake of high efficient content distribution.

To protect the content stored in the public cloud from being accessed by unauthorized parties, CloudSeal uses proxy-based data outsourcing scheme to ensure that only authorized subscribers are able to obtain the decryption keys. The content provider preprocesses cleartext locally by calling $\pi.KeyGen$, $\pi.SEnc$ and $Pro.PEnc$ functions described in the previous section and then publishes the processed content to the public cloud-based storage. The structure of outsourced data enables CloudSeal to protect content confidentiality as well as outsource flexible access control policies. To delegate different access control mechanism, the content provider distributes corresponding re-

encryption keys rk to the application service in the cloud. The application service transforms the ciphertext in the cloud by calling $Pro.ReEnc$ function with rk so that subscribers can decrypt them with the shared secret key uk. When an update happens in the user group, e.g., a user joins or leaves the group, the content provider updates the content re-encryption key rk stored in the application service to invalidate previous version of the content. Specifically, the content provider calls $Pro.ReKey$ to generate a new delegation key for the application service. Then the application service produces new ciphertext by executing $Pro.ReEnc$ on the outsourced cipher content in the cloud with the new re-encryption key rk. A straightforward way to distribute re-encrypted data with content delivery network is to invalidate previous cache content first and then distribute the new one. However, given a dynamic user group within the data sharing services in the cloud, this simple way will cause large distribution overhead. In order to take fully advantage of content delivery network and improve the efficiency of content delivery for data sharing services in the cloud, we design a novel content delivery strategy as shown in figure 4.3. In our new delivery strategy, the small part of the dual encrypted content *u* is stored in the cloud storage service and the main part $\langle v, w \rangle$ is cached in the delivery network to accelerate content distribution. During the re-encryption process, only a small part of the ciphertext of the content needs to be updated. The main part remains unchanged and can be persistently cached within the delivery network. When accessing a content, a client has to obtain all parts from cloud and distribution network in order to decrypt and render the content. This feature seamlessly achieves security objectives and efficient content distribution together.

Figure 4.3: CloudSeal content delivery strategy

4.3.2 User Management

For user management, CloudSeal has operations of `User Revocation` and `User Subscription`. CloudSeal supports a group of users' access to a set of encrypted content (or a channel) by sharing a secret key *uk* within the group. When a user joins the group, the content provider issues her shares of future secret keys as well as the current secret key. When a user leaves or is revoked from the group, the content provider broadcasts this user's share of the new decryption key to the entire group so that the remaining users are able to generate the new decryption key autonomously. Details are described as follows.

User Revocation happens when a subscriber leaves a group or is revoked by the content provider. It requires key revocation operations in the group. Our key revocation process is based on the k-out-of-n threshold secret sharing scheme. We consider the following two cases.

- *Case I:* There are $k-1$ users to be revoked at one time. The content provider revokes $k-1$ users with shares $P(x_1), P(x_2), ..., P(x_{k-1})$, respectively. The content provider broadcasts

the shares of secrets and identities of these users $< x_1, P(x_1) >, < x_2, P(x_2) >, ..., < x_{k-1}, P(x_{k-1}) >$ to the entire group. Each user x in the group combines her share of secret $< x, P(x) >$ with these $k-1$ shares, to interpolate the new secret key $uk' = P(0)$. For example, $k = 2$, $P'(0)$ can be calculated by $\frac{x}{x-x_1}P'(x_1) + \frac{x_1}{x_1-x}P'(x)$ according to Equation 2.1. The content provider uses uk' as the new shared secret key to generate re-encryption key for non-revoked users.

- *Case II:* There are *t* users to be revoked, where $t < k - 1$. The content provider performs the revocation by sending the t shares of secret and additional $k - t - 1$ shares of the secret of polynomial *P*. These additional shares are values different from any existing users.

Polynomial P is then removed from the list L. If the list L is empty, the content provider adds new polynomials, as well as computes and distributes corresponding secret shares to current subscribers (for future interpolation purposes).

User Subscription happens when a user joins a group. Successful subscription authorizes a user's access to protected content. To prevent new users from accessing content published before they join (for forward security), the key revocation process is required to be executed as follows.

- Upon receiving join requests from t new users, the content provider obtains the first polynomial P' on list L, and calculates key $uk' = P'(0)$; uk' is sent to the new users in secure channels.

- The content provider assigns each new user a unique identity $x_i \in \mathbb{Z}_r$ and her share of secret from polynomial $P'(x_i)$, along with x_i's values from the other polynomials on list

L. The content provider sends these polynomial values, except for $P'(x_i)$, to the new users respectively for future key updating through secure channels.

- The content provider broadcasts new users' share of secret $< x_i, P'(x_i)$ to the current group members for new key generation. If $t < k-1$, the content provider generates $k-t-1$ more share of the secret with P' and sends them to the entire group. These $k-t-1$ shares of the secret are different from any existing values. P' is removed from the list L.

For each current group member x_j, upon receiving $< x_1, P'(x_1) >, < x_2, P'(x_2) >, \ldots, < x_{k-1}, P'(x_{k-1})$ from the content provider, she calculates the new key with her share of secret $P'(x_j)$ for P' that was received when x_j joined earlier. This user can recover the new secret key $uk' = P'(0) = b$ by calculating $P'(0)$ with its share and received share.

4.3.3 Security Analysis

CloudSeal aims to protect content confidentiality and user access control (forward and backward security). We briefly discuss them next.

Content Confidentiality

For content confidentiality, CloudSeal achieves it through our dual-layer encryption algorithms. Details of the security analysis of our scheme is presented in section 3.2.4.

Forward and Backward security

CloudSeal is designed to protect content forward and backward security. When a user joining or leaving event happens in a group, the content provider issues a new re-encryption key for this group to the application service, then requires the application service to alter the content to be delivered with the newest re-encryption key, and finally updates the entire group with the new group information. For a join event, the content provider sends her the latest decryption key and her shares of secrets for future key update. Consequently, a new user cannot decrypt the old content with the new decryption key, and a revoked user cannot decrypt new content with her old keys. Note that CloudSeal does not prevent a user from sharing decrypted content or decryption keys to unauthorized users. A digital rights management tool, such as the Microsoft DRM component in Netflix [105], can be used to solve this problem, which is out of the scope of CloudSeal.

Collusion Resistance

Collusion between delegatees and the proxy is one weakness of many proxy-based re-encryption algorithms. The goal of collusion resistance in such systems is to prevent the recovery of a delegator's secret keys by combination of the issued re-encryption key and delegated decryption keys. CloudSeal utilizes the proxy-based re-encryption scheme proposed in [28], which has been proven to be collusion resistant. This guarantees that the secret key of a content provider is secure even with a user or the cloud provider obtains both re-encryption key and the user's decryption key. With this algorithm, CloudSeal ensures that the entire outsourced content cannot be compromised and

the content provider preserves the control of security policies by issuing of different re-encryption keys to different authorized groups of subscribers.

Besides security properties for content sharing and delivery via public cloud, CloudSeal uniquely achieves content distribution efficiency. When a group state changes, only a small part of the cipher content needs to be re-encrypted in cloud, while most of the content objects can be cached in the delivery network and shared by users. The separation of content operations (data plane) and user management operations (control plane) further enables flexible and scalable deployment of CloudSeal in the public cloud and network environment.

4.4 Implementation and Evaluation

In this section, we first present the implementation of CloudSeal with Amazon Web Services (AWS), and then evaluate its system and network performance.

4.4.1 Implementation

Content sharing applications with public cloud. We spent nontrivial efforts to build a simulated content sharing service based on three AWS services [3]: Elastic Compute Cloud (EC2) service to host an application service for subscribers, Simple Storage Service (S3) to store the cipher content for content providers, and Content Delivery Service (CloudFront) to distribute content. Amazon EC2 provides a virtual environment and allows developers to launch virtual machine instances with

various operating system and applications. Amazon S3 is a cloud-based storage system, which stores data in *buckets* and allows write, read, and delete operations on them. Amazon CloudFront provides high-speed content delivery service by automatically caching content in the nearest edge locations and delivering the cache to end users. To achieve security enhancement of this service, we extend its functionality with algorithms and protocols aforementioned in CloudSeal.

We implement the application service as a website written by PHP and host it with an Apache server running in an Amazon Linux EC2 instance. In order to directly store and manipulate the content in S3, we create a `connection object` with Python library `boto` [4], which calls the function *boto.connect_s3(s3.key, s3.ID)* to establish connections between the EC2 instance and S3 storage. *s3.key* and *s3.ID* are the key and identifier number of Amazon S3 instances assigned when created.

Our content is stored in the buckets of Amazon S3, which assigns each bucket with a global unique name and each file a URL composing with bucket name and file name. We use this URL as a content object in the application service web page. For example, file *video1.mp4* is stored in bucket *bucket1*, the access URL to that file is `http://s3.amazonawa.com/bucket1/video1.mp4`. Applications can use HTTP or BitTorrent-like protocols to access data stored in Amazon S3. Our application service uses HTTP protocol. Users can request files by file names from the file list stored in the application service with `HTTP GET` request. For each published content, we create two buckets in S3: one is for the originally published cipher content ¡u_{SK}, w, v¿ from the content provider, which is not updated once stored, and the other for the cipher content ¡u_{uk}, w, v¿ to be delivered, which is updated once a user joins or leaves. Although both buckets can be made

publicly readable to users, we set `private` permission to the original published one since only the content provider and the application service need to access them. The other content bucket is `publicly` readable to all users. To avoid storage redundancy, only u_{SK} of the cipher content is stored, as the other part ¡w, v¿ is the same as the public bucket. This part is so small that we cache them on EC2 to facilitate the cryptographic operations.

CloudSeal with its cryptographic tools. We implement aforementioned cryptographic algorithms ($\pi.KekGen$, $\pi.SEnc$, $\pi.Dec$, $Pro.PEnc$ $Pro.ReKey$, $Pro.ReEnc$, and $Pro.Dec$) based on cryptographic functions from the OpenSSL library [11] and the pairing-based cryptographic library (PBC) [12] with independent native processes. We choose Advanced Encryption Standard (AES) with 16 bytes key as our symmetric encryption algorithm and implement the `CFB` mode of `AES` to randomize the ciphertext of each data block of a file. For pairing-based encryption, instead of mapping an arbitrary M to a certain field $\mathbb{G}_T$, we map the elements in $\mathbb{G}_T$ to byte array in C language and use `XOR` operation in our implementation to produce w and v. This approach accelerates the encryption operation for the content provider. For key updating, we choose random linear polynomial formula so that only one user can be revoked with a single re-encryption operation.

We further develop a web application as the administrative service to assist the content provider for the user and key management on EC2 instance. This service enforces the cryptographic tools running on the same EC2 instance as the application service. When a user signs up in a group with the administrative service, the service updates current decryption key of the group and assigns this key and a share of future decryption keys to the new user. It updates the entire group with the new user's share. When a user decides to leave the group, the user sends `Leave` message to the

administrative service. The service again updates the current decryption key and distributes the share of secret of the revoked user. When an update happens in a group, the administrative service blocks any new file downloading activities in this group. Then it calculates the new key and distributes the revocation information to the remaining users, and then re-encrypts the content and stores it back on EC2 with the newest decryption key. After this, this group can start downloading files. In order to keep the up-to-date decryption key, the user periodically checks the administrative service to see whether or not there is a revocation by sending `Update_Checking` HTTP requests to the server. Once there is a revocation, remaining users send `GET_Update_Info` packets to get the newly revocation information and update her decryption key accordingly. As we integrate the administrative service into the HTTP server of the application service to measure the key and user management mechanisms, the user needs to actively pull the update information from the service, instead of passively receiving message from the service. We plan to realize the administrative service with an XMTP (XML MIME Transformation Protocol) server in the future, with which users can receive push notification from the server.

4.4.2 Experimental Evaluation

To confirm that CloudSeal achieves the performance objectives (mentioned before, we sought to answer the following questions to demonstrate that the security mechanisms from cryptographic algorithms in CloudSeal bring acceptable costs to end-to-end performance.

1. What is the overhead of cryptographic operations including content encryption by the content

provider and the content decryption by the subscribers?

2. What is the overhead of the re-encryption operations including content re-encryption by the application service in the cloud? Will the application service become performance bottleneck due to the overhead?

3. What is the affect of content delivery network to the cloud-based content distribution?

We have conducted a number of experiments to evaluate CloudSeal in the system and cloud levels in May, 2011. We examine the host-based efficiency of the cryptographic algorithms with different pairing types and the user management costs for communication and storage. We further conduct experiments on Amazon public cloud environment to evaluate efficiency of different cloud services of CloudSeal with two EC2 types (small, and medium) and different data distribution mechanisms (with or without CloudFront).

Efficiency Evaluation

Computation Cost. To show the performance at the content provider side and the subscriber side with cryptographic operations, we conduct content encryption (by content provider) and decryption (by content consumer) tests locally on a desktop with Intel Duo CPU 2.93GHz, 4GB RAM, SUMSUNG 7200RPM, and CentOS 2.6. The encryption time consists of symmetric encryption with CFB mode, pairing operation, and `XOR` operations. The decryption time includes key generation, symmetric decryption, pairing operations, and `XOR` operations. We choose two symmetric pairings from the pairing based library [12], including Type E pairing *e.param* and Type A pairing

Figure 4.4: Overhead of encryption operations with different pairing types.

Figure 4.5: Overhead of decryption operations with different pairing types.

a.param, to examine the impact from different pairing types to CloudSeal according to the scheme of CloudSeal. The symmetric key length is 16 bytes. The length of pairing-based master key and decryption key is 20 bytes for both pairing types.

We put 9 different sizes of files in the desktop and run the encryption and decryption algorithms 20 times on each file to compute the average processing time. As shown in Figure 4.4 and Figure 4.5, encryption and decryption time increase along with the content size, while there is a tiny difference between the two symmetric pairing types. For content decryption at consumer side, it takes less than 30 seconds to decrypt a 800MB video file.

Communication Cost. We investigate the communication cost caused by a group update, e.g., a user joins or is revoked. Recall that the degree of the polynomials is the number of users who can be revoked simultaneously. For example, a degree 2 polynomial implies that the content provider can revoke two users at one time. Assume we choose a degree d polynomial and we need B bytes to send one user's revoking information, if we have r users remain in the group, each revocation causes $d * B * (r + d)$ bytes traffic in the network with our broadcasting mechanism, where $(r + d)$ is the total number of users in the group. Suppose there are 100 users in one group, d equals to 2, and B is 40 bytes according to our implementation, the total amount for one update in CloudSeal is around $8KB$.

Storage Cost. We further look into the storage cost for the key management at both content provider side and subscriber side. In order to sustain system state, the content provider needs to keep current user secret key and a list of polynomials in terms of sets of coefficient for future system update. In our implementation, 20 bytes are enough to represent user secret key or each coefficient of polynomials. The average amount of storage cost for each group at content provider side is 6KB given 100 polynomials of degree 2. This cost is small enough to support a large number of groups of subscribers in the service. For subscribers, they only need to keep their own values of the list of polynomials with total amount of $2KB$ for 100 polynomials of degree 2.

Application Service Performance

We evaluate Amazon EC2 service on two different instance types: small and medium, located in AWS North California data center. The small EC2 instance consists of one core CPU with 1 ECU

Table 4.1: Re-encryption Time (Seconds) of 600MB Content on Different Amazon EC2 Instances.

Pairing	Key Length (Bytes)	Small EC2	Medium EC2
TypeE	20	0.00007425	0.000071
TypeA	20	0.00007925	0.000077

and 1.7GB memory, and the medium EC2 instance has two core CPU with 5 ECUs and 1.7GB memory. An ECU provides the equivalent CPU capacity of 1.0-1.2 GHz 2007 Opteron or 2007 Xeon processor. We focus on examining the re-encryption algorithm efficiency on Amazon EC2, which is the only operation on cipher content by the application service. We run each experiment 20 times and report the average running time of pure proxy re-encryption operations on 128 bytes data of each files.

As Table 4.1 shows, the operation on the small EC2 instance is slightly slower than that on the medium instance, which implies that computation ability of CPU has positive impact on cloud performance for our application services. Comparing with previous encryption and decryption time shown in Figure 4.4 and Figure 4.5, the re-encryption time on same size of content is significantly shorter. In real application, the execution time for proxy re-encryption operations might be larger than the data shown in the table 4.1, which includes extra I/O time for the file operations.

CloudSeal can serve multiple groups, each having a different set of authorized users. As a centralized component, the application service can be a bottleneck for performance, especially for the content re-encryption operations for all groups. Suppose there are 100 similar groups, each with

Figure 4.6: Number of required re-encryption operations with different churning rates

maximum 100 subscribers and 1000 content objects. We adjust the churn rate of a group to estimate the required re-encryption operations of the application service. The churn rate varies from 100 users per hour to 400 users per hour, which indicates the number of subscribers who join or leave the group per hour. From Table 4.1, we can calculate that, with the small EC2 instance, the application service can handle at most $1/0.00007425 = 13468$ re-encryption operations per second and with the medium EC2 instance, the application service can perform $1/0.000071 = 14084$ operations per second. Figure 4.6 demonstrates that with either small EC2 instance or medium EC2 instance, the number of re-encryption operations per second performed by the application service is much larger than the required operations per second along with different churn rate in the system. This indicates that the application service is adequate in providing revocation induced content re-encryption operations for groups for large churn frequencies. Furthermore, with elastic computing resources, the content provider can allocate more computing instances for the application service in an on-demand manner. Therefore, re-encryption operation in cloud will not be a bottleneck to the system's performance. Besides, as elapsed time for re-encryption operation is unrelated with the size of content, frequent revocation will not add large overhead on our system.

Figure 4.7: Downloading time for content delivery with storage center at Tokyo.

Figure 4.8: Downloading time for content delivery with storage center at N. California.

Content Delivery Efficiency

For content delivery network performance, we evaluate content delivery time with Amazon CloudFront. Four elements can affect the delivery efficiency: whether the CloudFront service is enabled, the locations of end consumers, deployed CloudFront edge servers, and the location of the content storage center. In our experiment, we store our content in two different Amazon S3 data centers: North California, USA and Tokyo, Japan. To initiate content delivery action, we develop a customized client with Python to continuously request files from one bucket stored in Amazon S3. We run 5 clients at North California, USA to leverage cache in local edge location. Each client requests 20 files and we measure the time of downloading all of them. As shown in Figures 4.7

and 4.8, without using CloudFront, the content delivery time in North California is much smaller than that in Tokyo. When the CloudFront is used, the delivery speed for content stored in Tokyo can be significantly improved by almost 10 times, while there is no obvious improvement for that in North California. We conjecture that this is because our client application is close to the North California S3 data center. Therefore the download speed does not change much when CloudFront is used.

Comparing the time for content delivery with the time of cryptographic operations (encryption, decryption, and re-encryption) shown in Figure 4.4, Figure 4.5, and Table 4.1, the cryptographic time is at least 40 times faster than the content delivery time. Therefore, the security mechanisms implemented in our prototype bring accesptable overhead.

In summary, our evaluation presents the efficiency of CloudSeal in content processing and content delivery. At subscriber side, CloudSeal brings acceptable decryption time. The overhead of CloudSeal does not affect user's experience. At the cloud side, the application service is not the bottleneck of the system and supports the efficient content re-encryption operations. Our system performs well when scaling up to a large number of subscribers. CloudSeal preserves the efficiency of content distribution of the content delivery network with a smaller overhead for cryptographic operations.

4.5 Chapter Summary

We designed and implemented CloudSeal, an end-to-end content confidentiality protection mechanism for large scale content storage and distribution systems over the public cloud infrastructure. By leveraging advanced cryptographic algorithms including symmetric encryption, proxy-based re-encryption, threshold secret sharing, and broadcast revocation, CloudSeal addresses unique challenges of efficient cipher content transformation, cipher content caching in the delivery network, and scalable user and key management. Through the prototype implemented on Amazon EC2, S3, and CloudFront, our experimental evaluation demonstrates that CloudSeal achieves the efficiency and avoids possible performance bottleneck. For future work, we plan to extend our current design to support an open service, where each user can publish content and delegate group membership control with our broadcast revocation library.

Chapter 5

Secure Data Processing Service with Vulnerable Cloud Virtualization Environment

Data protection in public cloud remains a challenge problem, especially for outsourced *data processing* in vulnerable cloud platforms that suffer from cross-VM attacks, such as side-channel attacks for cryptographic keys. We design and develop CloudSafe, a general and practical solution by integrating *cryptographic and systematic mechanisms* seamlessly to address this special issue. CloudSafe first allows a data owner to encrypt its data before storing it in the cloud storage. Upon the access request from an authorized cloud application running in a customer's virtual machine, a cloud-based proxy re-encrypts the data before it can be accessed by the cloud application. To

minimize the risk of key leakage in the cloud application side against cross-VM side-channel attacks, the final data decryption key is one-time and retrieved from the data owner in on-demand manner, such that any key leakage after the authorized access cannot compromise the data confidentiality. To further authenticate that only the authorized application can obtain the decryption key, we leverage a trusted agent of a virtualized platform to obtain the key and forward it to the virtual machine. We present details of CloudSafe in the remaining of this chapter.

5.1 Problem Description and Motivation

With rapid growth of cloud applications, individuals and enterprises intend to use various cloud-based services to store and manage personal or commercial data, ranging from hundreds of megabytes photos and music of an individual to gigabytes transaction and customer data of a company. Meanwhile, numbers of cloud-based applications provide more and more computational and management services for individual users and enterprises to process their data stored in cloud. With the rise of cloud-based data processing services, cloud users are encouraged to focus on the innovation of their own business without burdens from development and maintenance of underlying hardware and software. This newly born business model, however, cannot be exploited thoroughly without considering security and privacy issues it brings in. For one thing, commercial data contains sensitive information of an individual or a company and the cloud that stores and processes the data may be *honest but curious*. Sensitive data cannot be released to the remote untrusted environment without strict access control policies. For the other thing, users who rent

data computational or management services from the cloud are obliged to protect their own data in agreement with SLAs [19]. The cloud providers are able to flee from liabilities on data protection when security breach happens [19].

A common view to protect data confidentiality in the cloud is to apply cryptographic techniques on the data before it is outsourced to the untrusted cloud environment. However, how to efficiently control the access to those encrypted data and enhance their utility is still a problem. In response to this problem, researchers have proposed various solutions. KP-ABE [63] and CP-ABE [75] algorithms assign attributes to encrypted data and provide keys with access structure to users who can access these data. The comparison on those attributes and access structures can decide whether or not a user is able to access the data. Due to their complexity, these techniques usually work on protecting data encryption keys only, which in turn protects the target data. As a result, they are under attack after the data encryption key is leaked, e.g., by cloud-resident applications. Homomorphic encryption makes computation directly on encrypted data possible. However, current homomorphic encryption algorithms are not practical enough for general computation operations. Those security concerns are not only among academics, many cryptographic techniques have already been used in current commercial products. For example, Dropbox [6] stores and synchronizes the ciphertext of its customers' data with Amazon S3 storage systems via AES-256 encryption and SSL separately. However, the data encryption key is controlled by Dropbox service, instead of the data owner.

Moreover, malicious entities in the application domain, which try to steal secret decryption keys from trusted but buggy cloud application and its system environment (virtual machine), cannot

be ignored. Software vulnerabilities have been identified in the rented virtual machines of public clouds such as Amazon EC2 [31], where 98% of Windows AMIs (Amazon machine image) and 58% of Linux AMIs contain software with critical vulnerabilities, mainly due to the fact software running on each AMIs are often out of date and not patched in on-time manner. Trojans and spyware have been found in many AMIs also [31]. Some AMIs have backdoors installed. These vulnerabilities allow variant runtime attacks, e.g., via code-injection or return-to-libc. On a virtualized cloud platform, malicious applications in one domain may attack applications in another domain, e.g., through underlying virtualization layer with side channel analysis [108, 124]. Yet another type of threats comes from offline VM image analysis. For example, many AMIs have been found to leave application and authentication credentials permanently [31, 46]. Malicious cloud administrators can easily leak these data to unauthorized parties.

Designing effective cryptographic techniques seem to be a promising direction to alleviate those issues. Many directions have been explored. For example, FHE (fully homophobic encryption) allows computation to be applied directly on encrypted data and returns the same results as that on plaintext of the data. Since the data always stays in its encrypted form when rest at the other parties, the data owner no longer needs to worry about the security of her outsourced data. However, due to its computing limitation and complexion, the practicality of FHE is still not clear now. In order to utilize computing resources in the public cloud, we need to execute decryption operations on the outsourced encrypted data in the public cloud environment.

FDE (fully disk encryption) is another type of solution, which relies on encryption and key management mechanisms to ensure the confidentiality and access control to the outsourced data. This

technique is able to solve the data security problem in the cloud to some extent. But its own security, besides unsatisfied performance in the large-scale environment, is threatened by the attacks from insecure cloud virtualization environment. Cross-VM side channel attack is able to steal the encryption/decryption keys when they are using in a victim virtual machine; misconfigurated virtual machine images subtly ship sensitive information to the outside. It may be easy to build a clean and secure virtual machine in the cloud from scratch to avoid potential security risk harbored by public shared virtual machine images, it is difficult to escape from eavesdropping through hidden side-channels set up by malicious neighbors on the same platform. To our best knowledge, the solutions against this side-channel attack are not effective let alone defending all the potential sid-channels existing the cloud. Cryptography alone is not enough to provide data security. More security mechanisms are expected to provide data security in the public cloud environment.

In this work, we introduce *CloudSafe*, a new approach that offers security means for a data owner to secure and control her encrypted outsourced data in the public cloud with the presence of the semi-trusted cloud service providers and malicious cloud customers. In addition to previous cryptographic-centric solutions with merely focus on outsourced data security in the cloud, CloudSafe stresses new security issues arose from cross-VM side channel attack against the virtualization environment of the public cloud. This additional emphasis enables CloudSafe to succeed in both protecting outsourced data security and surviving against the vulnerable virtualization environment in the current public cloud environment.

To fight against cross-VM side channel attack in the public cloud virtualization environment, we observe that the stealing behavior involving in such attack can only be accomplished if the secrecy,

such as data decryption keys for AES algorithm, is loaded into the victim machine's physical component, e.g. CPU cache. With this observation, a simple solution to conquer such attack is to use one-time secret keys in the victim machine: each secret key is discarded once it is been used. However, this approach brings two major challenges: 1) complicated key management due to the large amount of one-time data decryption keys; 2) heavy computation requirements over encrypted outsourced data by the data owner with frequent updating data decryption keys. Without overcoming this two challenges, it is impractical to use one-time key in the current public cloud environment.

In response, CloudSafe propose two techniques to overcome the mentioned challenges. Firstly, CloudSafe defines a flexible and efficient cryptographic scheme to support one-time data decryption key strategy with small cryptographic computation requirement and few extra storage cost. This scheme ensures that the outsourced data not only is confidential when storing in a public cloud storage but also can be efficiently adjusted according to different one-time data decryption key without heavy computation required on the data owner side. Particularly, CloudSafe designs a general cryptographic model to process data and support one-time key strategy. This general model can be realized efficiently with existing solutions.

Moreover, CloudSafe further proposes a novel centralized key distribution framework inside the public cloud environment to assist storing and distributing one-time data decryption keys in the cloud. To store a shot-lived data decryption, CloudSafe builds a centralized key management framework across a data center to securely store the one-time data decryption key on behalf of the computing entities . To distribute the one-time data decryption key from the data owner to

the computing entities in the cloud, CloudSafe splits it into two phases with the assistance of the framework. The first phase is to distribute the key into the the framework in the public cloud data center where the computing entities is running over public network; the second phase is to distribute the key from the the framework to the computing entities through the intranet of a cloud data center. Through the entire process, the key management framework guarantees two things: 1) the data decryption key is securely stored to the computing entities; 2) the computing entities is able to access the key from anywhere in the data center. These features enable secure data decryption key usage in a vulnerable virtualization environment and flexible support of dynamic virtual machine migration activities in the public cloud.

5.2 Models and Design Goals

Before dive into details of CloudSafe, we review the prerequisite setting for other parts of this chapter in the following subsections.

5.2.1 Problem Scenario

A popular usage of public cloud service is called IaaS (infrastructure-as-a-service). IaaS offers general computing resources to its tenant in the form of virtual machine images, storage blocks, deliver network, and so on. Customers can build its own web business by renting those infrastructures. For example, the data owner is able to build an online gallery with those infrastructures. She can store her digital painting to the Amazon cloud storage and ran an web application by Amazon

EC2 services to show those paintings to the website users. The web application only retrieve data from the cloud storage without writing back. As shown in figure 5.1. Three players are involved: infrastructure service provider, cloud customers, and virtual servers.

- *Infrastructure service provider:* also referred as the *cloud provider*, carries infrastructural resources including physical storage devices, computing machines, and network equipments, and provides virtualized resources as services to cloud customers.

- *Cloud customers:* purchase infrastructure services on demand. A cloud customer who stores and processes her data in the cloud is also called the *data owner*. A cloud customer who consumes data stored in the cloud is called the *data user*. Those data users might be malicious and perform cross-VM side-channel attacks, and they are referred the *malicious cloud customer* or *attacker*.

- *Virtual servers:* are built by a cloud customer on virtual machines leased by the *cloud provider*, which host various applications for data processing.

A data owner stores her data in the cloud storage provided by a cloud storage provider and builds several virtual servers to process the data by renting virtual machines hosted by a cloud computation entity provider. The cloud provider, as the infrastructure service provider, owns the leased physical devices, e.g. storage disk and physical machines, and maintains them for their customers. The cloud customer has complete control over the rented virtual machines and the storage space, but she has no access to the lower layer of the beneath infrastructures such as hypervisors and storage disks. She may rent different infrastructure resources from one cloud provider or multiple

Figure 5.1: The problem scenario of CloudSafe

cloud providers. Since the cloud customer doesn't possess the rented physical devices, she may want to export her data in encrypted form instead of plaintext to protect its confidentiality; she may also want to control the access to her data and only grant permissions to authorized virtual servers in the cloud.

The cloud provider may arrange multiple cloud customers to share the same physical infrastructure to maximize the utility of her resources without cloud customers' awareness. For example, multiple virtual machine images might be running on the same physical machine in Amazon EC2 cloud. Amazon EC2 administrator migrates virtual machines images from one machine to another and turn on and off a physical machines to optimize the usage of the physical machines and save the energy.

Infrastructures leased by the data owner may belong to different data center in different locations, for example, a cloud provider may have multiple data center in different locations, or the data

owner may rent the infrastructure from several cloud providers with their data centers in different location. Data may be moved between data centers. In this paper, we are tolerant to the separated locations of cloud storage and virtual servers: they can belong to different data center. However, we assume the same infrastructure resources are from the same data center.

The data owner is the only source to provide data to the *virtual servers* in the scenario. Although it is possible that a data processing entity consume data from different origins, we treat our model with a single data owner and regulate the direction of data flow only from the cloud storage to virtual servers for simplification purpose. The scenario of multiple data owner and bi-directional data flow, which might introduce more security issues, will be our further interests.

5.2.2 Threat Model

We consider two types of adversaries: semi-honest cloud providers and malicious cloud customers. A semi-honest cloud provider may illegally access to the outsourced data stored or distributed over her physical machines; the malicious cloud customers is able to perform cross-VM side channel attacks in a public cloud. By this attack, they can steal sensitive information, such as data decryption keys, from a virtual server running on the same physical machine. While it is possible for malicious cloud customers to steal data by sharing a misconfigured virtual machine images in the public images sharing pool, we instead focus on side-channel attacks by malicious cloud customer and leave this problem to the future research. For the former attack, we focus on protecting encrypted data from illegitimate access by unauthorized parties. While there is a chance

that the plaintext of encrypted data will appear in the cloud environment for computation purpose, for example, a virtual server needs to decrypt the data it obtains from the cloud storage before compute it, we do not intend to prevent stealthy behaviors on such data. This is because illegal access to the encrypted data in the storage is more dangerous than the illegal access to the plaintext on data processing entities when considering the amount of data that are affected. We focus more on encrypted data security during storage and distribution phase in this paper.

Cloud customers who require confidential cloud services are our victims and fully trusted. We also trust the quality of services offered by the cloud providers, by which the data are intact either be stored in a storage or be distributed in the cloud according to SLAs. In addition, we consider the underlying infrastructure in the cloud including hypervisors and the hardware are not compromised. This means that we don't consider attacks against hypervisors and any insider threat or abuse of cloud administrative rights within a cloud. Specifically for Xen virtualization environment [15, 32], malicious attackers in the domU domain can not penetrate into dom0 domain on a machine, that is, a malicious cloud customer can only mount cross-VM side-channel attacks among virtual servers in different DomUs on the same physical machine.

Traditional threats on a common computer, such as software vulnerabilities, malware, and malicious network activities, exist in cloud environment as well. Those type of treats are well-known and also very important for data security not only in the cloud but also on personal computers. We are aware those security issues in this paper and treat the virtual server a insecure place to store sensitive information. However, the solutions to protect the virtual server from those vulnerabilities is not in the scope of this paper.

Besides the security goals, our system should be efficient and flexible enough to be applicable in a real system. Since the encrypted outsourced data has to be transformed frequently according to the current one-time data decryption key, the operation time should be acceptable. We achieve this by adapting an efficient cryptographic scheme and utilizing a practical key management framework.

5.3 Design Overview

In this section, we first demonstrate two initial attempts to solve the mentioned security problems existing in current cloud environment. We then present the overview and main operations of CloudSafe in details.

5.3.1 Initial Attempts

Simple one-time data decryption key. A straightforward solution to protect cryptographic key usage is to adopt one-time data decryption key on original data directly, meaning that a one-time key is used to encrypt data before it is outsourced in cloud. After being used once by a virtual server, the key is revoked. This requires the data owner to securely delete the current encrypted data in the cloud, re-encrypt the original data, and then upload the new ciphertext data to the cloud again. Obviously this solution is impractical when frequent and large-scale data decryption operations are needed in the cloud.

Hypervisor-based key usage. In order to avoid heavy computation and large network overhead

caused by the simple one-time data decryption keys, an alternative solution to protect cryptographic key usage in virtual servers is to adopt hypervisor-assisted key usage. According to our threat assumption, cross-VM side-channel attacks happen only between virtual servers on the same physical machine, and it is hard for a malicious cloud customer to build side-channels to the hypervisor and privileged domain. Therefore, moving data decryption operation from a virtual server to the hypervisor thus using long-term data decryption key in the hypervisor is secure enough to protect cryptographic key usage in the cloud virtualization environment, since the key never resides in the vulnerable virtual server environment. However, with the increasing number of virtual servers that are running on the same physical machine, usually from different cloud customers, it is discouraging to execute computation-intensive operations in the hypervisor or privileged domain for performance consideration. Furthermore, dynamically loading guest virtual server's code for data decryption purpose in privileged domain may introduce new risks to virtualized platforms.

5.3.2 Design Strategies

To successfully protect outsourced data against unauthorized access and securely distribute and use data decryption key with the virtual servers against malicious cloud customers, CloudSafe adopts the following strategies to overcome the shortcomings stated above:

Combination of one time decryption key and proxy re-encryption. The method of simply using one time data decryption key exposes heavy-weight computational cost on data owners. In order to leverage the cloud computational resource and facilitate one time data decryption key, we employ

the idea of proxy re-encryption: Given a master key, the data owner generates a key pair having specific purpose, one of which, called one time re-encryption key, is distributed to the cloud so that the cloud can re-encrypt the ciphertexts to new ciphertexts, and the other one, called one time data decryption key, is issued to the data user so that the data user can decrypt those new ciphertexts. Note that the one time data decryption key can be used to decrypt ciphertexts re-encrypted with designated one time re-encryption key. That is, given $\langle \mathsf{reencryption\_key}_1, \mathsf{data\_decryption\_key}_1 \rangle$ and $\langle \mathsf{reencryption\_key}_2, \mathsf{data\_decryption\_key}_2 \rangle$, the data user only can use $\mathsf{data\_decryption\_key}_1$ to decrypt re-encrypted ciphertext under $\mathsf{reencryption\_key}_1$, rather than those under $\mathsf{reencryption\_key}_2$. This strategy enjoys the following advantages:

- The data owner can offload the heavy-weight computational cost to the cloud by leveraging the elastic computational power provided by the cloud. In addition, without uploading new ciphertexts, it dramatically reduces the bandwidth overhead of key revocation process.
- The data owner can directly implement access control on its demands, e.g. by controlling the distribution of one time data decryption key and re-encryption key.

Centralized key distribution framework in cloud. In order to securely distribute keys to corresponding entities in public cloud, CloudSafe adopts different channels for different keys. First, we deploy the proxy in a dedicated cloud instance, which only runs a single virtual machine of the data owner. With this, the cross-VM side-channel attack is not a threat and the proxy can have a public/private key pair to build secure channel with the data owner. Therefore the re-encryption key can be distributed with this channel securely. For data decryption key, since it travels through pub-

lic in-cloud network, and we cannot assume a secure channel between a virtual server and the data owner due to cross-VM side-channel attacks in the virtual server side, we develop a centralized key distribution framework by leveraging the relatively secure hypervisor and Dom0 environment on each cloud platform. With this, a secure channel can be built between the data owner and a trusted agent in Dom0, which in turn transfers data decryption keys to the local virtual server without going to public in-cloud network. This mechanism achieves high assurance of key distribution in public cloud environment.

5.3.3 CloudSafe Overview

To embody these design strategies, CloudSafe consists of three main operations: outsource operation, authorization operation, and distribution operation. The first two operations are executed by a data owner with a cryptographic scheme specified for CloudSafe to protect outsourced data confidentiality, and the latter one is carried out by the key management framework in CloudSafe to guarantee secure distribution and usage of data decryption key by a virtual server in a vulnerable cloud environment. The overall workflow of CloudSafe is depicted in figure 5.2. Next, we look into details of each operation.

Outsource operation. An outsource operation provides confidentiality to the outsourced data stored in a cloud storage. The operation is executed by a data owner when she needs to export her data to a cloud storage. During this operation, the data owner takes the original plaintext of her data into the encryption cipher with a secret key and outsourced the encrypted data to the cloud

Figure 5.2: CloudSafe workflow

storage. The data owner chooses the secret key for each outsourced data and records them locally for the authorization usage. This operation is done only once for each data (e.g., a file) before it is exported from the data owner to the cloud storage. The outsource operation achieves the following properties:

- The outsource operation provides a common encryption functionality that can be used on different format of data.
- The outsource operation ensures that each encrypted outsourced data can be efficiently transformed so that the data owner is able to authorize it with different data decryption keys.
- The outsourced data is independent to each other so that decryption keys to one data cannot be used to other data in the cloud storage.
- The encryption algorithm is lightweight, making ciphertext transmission less overhead com-

paring to the transmission of the plaintext of the data.

The confidentiality achieved by the outsource operation comes from the small granularity of encryption operations: each data are encrypted stored in the cloud storage and is independent to each other. As long as the outsourced data stays in its encrypted form, cloud provider can not peak its plaintext from the cloud storage. Even the cloud provider is able to *copy* a plaintext from a virtual server in the cloud, she cannot compromise the *same* data in the cloud storage let alone other data storing in the cloud storage.

Authorization operation An authorization operation prevents unauthorized access to the data in the cloud storage via one time decryption keys and re-encryption keys. When receiving a data request from a legal virtual server, firstly the data owner takes the data request information and generates a pair of one time keys $\langle \mathsf{reencryption\_key}, \mathsf{data\_decryption\_key} \rangle$, and delivers them to a proxy service and the virtual server, respectively. Then the proxy re-encrypts the data with the received one time $\mathsf{reencryption\_key}$ and sends the re-encrypted ciphertexts to the virtual server. Note here a pair of one time keys is generated for each data request, and CloudSafe ensures that only the re-encrypted data can be decrypted by an authorized virtual server in the cloud, leaving the rest of the outsourced data secure in the cloud storage. With efficient outsourced data transformation, CloudSafe is able to support one time key strategy with affordable computation overhead. Besides, the re-encryption operation offloaded to a proxy further reduces the workload at the data owner side. It has the following features:

- The authorization operation provides data transformation to each data request that can be

decrypted by the one-time data decryption keys received by the virtual servers in the cloud.

- The authorization operation supports flexible authorization policies. The granularity of authorization on the data in the cloud is fine-grained, making the exposure of plaintext in the cloud minimum.

- The authorization operation is independent to each data request so that unauthorized parties cannot access the data with expired one-time keys.

- The implementation is lightweight and the proxy is efficient enough to intercept the data flow from the cloud storage to the virtual server.

With the efficient outsourced data transformation guarantee, CloudSafe is able to support the one-time key strategy with affordable computation overhead. Besides, the data re-encryption operation can be offload to a proxy which further reduces the workload at the data owner side. With the pair of one-time keys generated in each data request, CloudSafe ensures that only the re-encrypted data can be decrypted by an authorized virtual server in the cloud, leaving the rest of the outsourced data secure storing in the cloud storage.

Delivering the pair of one-time keys is two different situation. To deliver the *re-encryption* key, since the proxy is built by the data owner and fully trusted, we can send the *re-encryption* key and related information of data directly to a proxy through a pre-built secure channel. However, it is not easy to build a secure channel with a virtual server in a vulnerable cloud environment due to the risk of key leaking. In response, CloudSafe designs an isolated distribution operation to distribute data decryption keys from the data owner to the virtual server, which is demonstrated next.

Distribution operation. A distribution operation provides secure distribution of one time key pairs with different channels. Since the proxy is built by the data owner and fully trusted, e.g., on a dedicated server in the cloud, the data owner sends the re-encryption key and related information directly to the proxy through a pre-built secure channel. The data decryption key is distributed through a novel centralized key management framework in the cloud. Firstly, the key is sent by the data owner to a key server through a secure channel over the public network; then the key is transfered from the key server to a requesting virtual server with the relay of the hypervisor on the same virtualized machine. The key distribution is operated every time when a virtual server sends a data request to the data owner. The distribution stage has the following attributes:

- The distribution stage provides a general secure key distribution solution against cross-VM side channel attacks in the cloud so that it can be used by other applications.
- The distribution stage isolates itself from virtualization environment in the cloud so as to maintain a secure place for the virtual server.
- The distribution stage is flexible enough to support different key usage patterns by the virtual server with different requirement on security and system performance.
- The key management framework sustains availability in a dynamic distributed environment, making the virtual server to be location-free to access the data decryption key over the framework.

Reliability during the distribution operation is also very important to a successful key distribution. To achieve this, the distribution operation relies more on the reliability provided by the cloud, such

as stable network and backup and recovery policies. The centralized key management framework can co-exist with other centralized cloud management tools, such as Xen server, to provide a centralized data-center wide services for the entire cloud. Using an isolated space to store the one-time data decryption key in a cloud environment makes it hard for attacker to steal secrecy from side-channels existing in the multi-tenant virtualization environment in the public cloud.

5.4 Cryptographic Scheme

5.4.1 Cryptographic Scheme for CloudSafe

Definition The cryptographic scheme in the CloudSafe consists of the following algorithms:

- $(\mathsf{mk}, \mathsf{pm}) \leftarrow \mathsf{Setup}(1^{\lambda})$: This is the bootstrapping algorithm run by a data owner to initialize the system. It outputs a master key mk for the data owner itself and the parameters pm which are made public.

- $\mathsf{cph} \leftarrow \mathsf{PEnc}(\mathsf{pm}, \mathsf{mk}, \mathsf{M})$: This is the data encryption algorithm run by the data owner before outsourcing M to the cloud. It outputs the encrypted version of M.

- $(\mathsf{deckey}, \mathsf{rekey}) \leftarrow \mathsf{KeyPairGen}(\mathsf{pm}, \mathsf{mk})$: This is the algorithm for generating a pair of one time keys, including a re-encryption key rekey, issued to a cloud-based proxy, and a decryption key, issued to an authorized virtual server where a data processing application runs. It is run by the data owner in order to authorize the virtual server to access the encrypted

data correctly. The pair of keys is distributed to the proxy and the virtual server via secure channels, respectively. Note that the decryption key deckey can only decrypt the re-encrypted ciphertext along with rekey.

- $\mathsf{recph} \leftarrow \mathsf{ReEnc}(\mathsf{pm}, \mathsf{cph}, \mathsf{rekey})$: This is the re-encryption algorithm run by the proxy to re-encrypt the stored ciphertext data to another ciphertext with the re-encryption key rekey. It outputs the re-encrypted ciphertext which will be delivered to the authorized virtual server.

- $\{\mathsf{M}, \perp\} \leftarrow \mathsf{Dec}(\mathsf{recph}, \mathsf{deckey}(\mathsf{cph}, \mathsf{mk}))$: This is the decryption algorithm run by an authorized virtual server with the decryption key deckey (or the data owner with the master key mk). It outputs the plaintext data M corresponding to the re-encrypted ciphertext recph, as well as the original ciphertext cph, or outputs error message $\perp$.

The security for this scheme has two-folds: (1) the master key cannot be leaked under the assumption that data users and the cloud do not collude, and (2) given one key of a key pair (a re-encryption key and a decryption key), the data user and the cloud cannot derive the other.

5.4.2 Construction and Discussion

Here we propose an efficient cryptographic scheme satisfying the desirable properties as above. Let AES be the standard AES encryption scheme, such that $\mathsf{AES} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$, where KeyGen is a probabilistic key generation algorithm, Enc is a probabilistic encryption algorithm and Dec is a deterministic decryption algorithm.

- $\mathsf{Setup}(1^\lambda)$: Given a security parameter λ, the data owner chooses a bilinear map $e : G \times G \to G_T$, where G and G_T are cyclic groups of the order p, an λ-bit prime. Let g be a generator randomly selected from G. Let H be a secure hash function, $H : \{0,1\}^* \to \{0,1\}^{\lambda_1}$, and F_k be a secure pseudorandom function, $F_k : \{0,1\}^{\lambda_1+1} \xrightarrow{k} \mathbb{Z}_p$ where λ_1 is another security parameter. The data owner selects a randomly from $\mathbb{Z}_p$ and lets

$$\mathsf{pm} = (e, g, G, G_T, p, g^a), \mathsf{mk} = (a).$$

- $\mathsf{PEnc}(\mathsf{pm}, \mathsf{mk}, \mathsf{M})$: The data owner selects s randomly from $\mathbb{Z}_p$. Let $\mathsf{M} = (\mathsf{M}_1, \cdots, \mathsf{M}_n) \in G^n$ uniquely identified by the file handler fid, and compute $x = F_{\mathsf{mk}}(H(\mathsf{fid})|0)$ and $y = F_{\mathsf{mk}}(H(\mathsf{fid})|1)$. The data owner encrypts M with AES, s.t.

$$\mathsf{M}' = (\mathsf{M}'_1 = \mathsf{AES.Enc}(x, \mathsf{M}_1), \cdots, \mathsf{M}'_n = \mathsf{AES.Enc}(x, \mathsf{M}_n)),$$

where x is the symmetric key for AES, then selects s randomly from $\mathbb{Z}_p$ and generates the ciphertext as

$$\mathsf{cph} = (g^{ys}, \{\mathsf{M}'_i e(g,g)^s\}_{i=1}^n).$$

- $\mathsf{KeyPairGen}(\mathsf{pm}, \mathsf{mk})$: Given the access request on data M specified by fid, the data owner selects t randomly from $\mathbb{Z}_p$, computes $x = F_{\mathsf{mk}}(H(\mathsf{fid})|0)$ and $y = F_{\mathsf{mk}}(H(\mathsf{fid})|1)$, and generates the decryption and re-encryption keys as

$$\mathsf{deckey} = (x, t), \mathsf{rekey} = (g^{t/y}).$$

- $\mathsf{ReEnc}(\mathsf{pm}, \mathsf{cph}, \mathsf{rekey})$: Given the re-encryption key rekey, the cloud performs the re-encryption on $\mathsf{cph} = (g^{ys}, \{\mathsf{M}'_i e(g,g)^s\}_{i=1}^n)$ as follows:

$$\mathsf{recph} = (e(g^{ys}, g^{t/y}) = e(g,g)^{ts}, \{\mathsf{M}'_i e(g,g)^s\}_{i=1}^n)$$

- $\{\mathsf{M}, \perp\} \leftarrow \mathsf{Dec}(\mathsf{recph}, \mathsf{deckey})$: Given the ciphertext $\mathsf{recph} = (e(g,g)^{ts}, \{\mathsf{M}'_i e(g,g)^s\}_{i=1}^n)$ and the decryption key deckey, the data user computes $X = (e(g,g)^{ts})^{1/t}$ and obtains the data $\mathsf{M}' = (\mathsf{M}'_1, \cdots, \mathsf{M}'_n), 1 \leq i \leq n$ by

$$\mathsf{M}'_i = \mathsf{M}'_i e(g,g)^s / X.$$

  Hence, the data user can decrypt M' with the symmetric key x and gets $\mathsf{M} = (\mathsf{M}_1 = \mathsf{AES.Dec}(x, \mathsf{M}'_1), \cdots, \mathsf{M}_1 = \mathsf{AES.Dec}(x, \mathsf{M}'_n))$. Similarly, given cph and the master key mk, the decryption can be done correspondingly.

Note that in the scheme each M is encrypted with a distinct secret key, determined together by the master key and the hash value of the data handler. This simplifies the key management since only the master key mk should be kept private. On the other hand, distinct secret key for each data can minimize the attack surface if the secret key corresponding to some data was compromised.

We also note that the one time decryption key deckey and re-encryption key rekey play vital role in enforcing access control policies for the data owner. First, only with one key from the key pair $(\mathsf{deckey}, \mathsf{rekey})$, neither the cloud nor data users can access any information from the stored data in the cloud. Second, given a key pair $(\mathsf{deckey}, \mathsf{rekey})$, deckey can only be used to decrypt the ciphertexts corresponding to rekey. This achieves the isolation between data users in the sense that one data user cannot decrypt the ciphertexts which is not designated for him. Through this, this scheme enables the data owner to realize flexible access control easily by generating and distributing $(\mathsf{deckey}, \mathsf{rekey})$. In what follows, we briefly analyze the security of the cryptographic scheme.

Security of the master key. In the above cryptographic scheme, we require that the master key and the data encryption key for each data can be kept privately. We note that the only way to compromise the master key mk is via obtaining x, y and breaking the one-way property of the pseudorandom function F. Given the assumption that F is secure, we can immediately draw the conclusion that the master key mk is secure for sure. On the other hand, even with either the data decryption key $\mathsf{Dec} = (x, t)$ or re-encryption key $\mathsf{rekey} = (g^{t/y})$, no one can derive the key y. Therefore, we can guarantee the privacy of the data encryption key because only cooperating x and y together can recover the plaintext data.

Security of the data decryption key and re-encryption key. We consider the case that either the cloud application with the data decryption or the cloud provider with re-encryption key is malicious but they do not collude together. Given only the data decryption key t of the key pair $((x, t), g^{t/y})$, we can see that the cloud application can only decrypt the re-encrypted ciphertext under rekey, but cannot derive re-encryption key $g^{t/y}$. Similarly, the storage provider cannot infer the data decryption key t only with the aid of re-encryption key $g^{t/y}$.

5.5 Key Management

To securely store and distribute one time data decryption keys from data owners to virtual servers, CloudSafe relies on a centralized in-cloud key management framework. In this section, we demonstrate the design of this framework within the context of Xen virtualization environment [15, 32] as Xen is a popular hypervisor that has been used in real cloud environments. Usually, a Xen vir-

Figure 5.3: The overlay of key management framework

tualized platform contains two types of domains: one Dom0 and multiple DomUs. The Dom0 is the control domain with high privileges, while DomUs are guest domains for virtual servers rented by cloud customers. As DomUs are insecure due to potential cross-VM side-channel attacks, we utilize Dom0 on each physical machine to enhance the security of data decryption key distribution and usage in DomUs.

5.5.1 Overview

Figure 5.5 depicts the overview of our key management framework, which consists of three main components: the key server, the global key storage, and multiple key clients.

The key server runs in a dedicated machine instead of a rented virtual machine in the cloud, e.g., a dedicate server provided by Amazon EC2 [1] or GoGrid [5], which run applications for a single cloud customer. It provides connection between the the data owner (or application running as the data owner) and the virtual servers rented by the data owner in the cloud. It has its own public/private key pair $\langle PK_{keyserver}, SK_{keyserver} \rangle$ to set up secure channel with the data owner. It

Figure 5.4: Cascade structure of the global key storage.

listens on certain ports to accept one time data decryption keys from the data owner and writes the received data decryption keys to the global key storage for future access from virtual servers. Logically only one key server is needed for a cloud. However, distributed key servers can be used for robustness and load balance.

The global key storage is a centralized data storage attached to the key server physically and remotely accessible by the key clients (virtual servers) over the cloud network. It stores one time data decryption keys on behalf of each virtual server in a cascade structure as shown in Figure 5.4. The `UUID` is the unique identifier of each virtual server in the cloud, which is generated when a virtual server is created by the hypervisor on a physical machine. As it is not changed during the entire lifecycle of a virtual server, we use it as the top folder name to identify the virtual server. The `Application` item is the identifier of a local program running in the virtual server that requires the data decryption key. The data decryption keys are stored in the `Key` folder with associated `Metadata` information of the encrypted data. Different versions of one time data decryption keys are sorted in different `Version` folders in order to state the usage order of them in the application. We use `Hash` to check the integrity of the value of the key.

A key client is a daemon program running in the Dom0 of a virtualized platform. It reads the data decryption keys from the global key storage on behalf of the applications running in the same virtual server. In order to use the key client, the applications that the data owner runs in a virtual server needs to call an API provided by the key client as shared library.

For security purpose, the global key storage is configured as read-only to key clients. Only the key server is able to write it. Besides, each key client is able to read the data exclusively assigned to the virtual server running on the same physical machine, e.g. a certain `UUID` folder.

To finish the tasks such as key write and key read operation during the key distribution, CloudSafe designs a set of protocols to support communication between the three components. The key idea behind these protocols is to protect the data decryption key from the vulnerable cloud virtualization environment while keeping the key available within a dynamic cloud environment. Next, we presents details of these protocols to demonstrate how CloudSafe achieve these goals.

5.5.2 Secure Key Distribution

To securely distribute a one time data decryption key from the data owner to a virtual server, we follow the protocol shown in Figure 5.5. Specifically, there are two key distribution phases. Phase one includes step 1 to step 4 shown in Figure 5.5, which delivers the key from the data owner to the key server through public network; phase two includes step 5 to step 7 shown in Figure 5.5, which dispatches the key from the key server to a virtual server via the key client through in-cloud network. Details of each step are described as follows. For sake of simplicity we ignore the

timestamps and nonce for preventing re-play attacks in the protocol messages.

Step 1: The virtual server sends a `data request` packet to the data owner with three fields: `Metadata` about the requested data, $\langle IP_{keyserver}, Port_{keyserver} \rangle$ of the key server information, and its own identifier `UUID` and application identity `Application`.

Step 2: Upon successful authorization verification based on the `UUID`, the data owner generates a pair of re-encryption and data decryption keys based on the `Metadata` she receives, according to the algorithm in Section 5.4.2.

Step 3: The data owner builds up a secure channel with the key server, and sends the data decryption key along with the virtual server identity, application identifier, `Metadata`, and key version to the key server through the channel.

Step 4: When the key server receives the key from the data owner, it writes the key into a directory named `UUID` of the global key storage, and sets the read permission only to the key client with `UUID`.

Step 5: The application in virtual server sends `key request` to the key client running in the same physical machine when it wants to use the data decryption key along with its identifier `UUID`, application identifier `Application`, and latest version information `Version`.

Step 6: The key client first verifies that the `UUID` is valid, and if so it finds the folder according to `UUID`, `Application`, and `Version` in the global key storage, and reads the data decryption key. Note that the key client only reads the folder with `Version` that is least greater than the received `Version`.

Figure 5.5: Key distribution protocol with CloudSafe.

Step 7: The key client sends back the data decryption key to the virtual server with the same `UUID` along with `Application` and `Version` information. The `Version` information will be used by the virtual server for next time key retrieving.

With these steps, the data decryption key never appears in the space of the virtual server before it is used. Since the data decryption key is one time for one time usage, this distribution protocol achieves high assurance that the virtual server will be the first one to use the key before any malicious cloud customers who steal the key through the side-channel when the key is using.

5.5.3 Virtual Server Lifecycle and Authentication

Another big advantage of the proposed key distribution strategy is its adaptivity to the dynamic cloud environment in which virtual servers are migrated from one machine to another and physical machines are on and off occasionally. The lifecycle of a virtual server has three stages: creation, migration, and termination. To authenticate and authorize virtual servers for key usage, and sustain key availability to the virtual server through its entire lifecycle, CloudSafe executes the following

steps.

Creation. Through certain interface provided by the cloud provider, the data owner creates a new virtual server instance in the cloud, and obtains a valid `UUID`. When the key server receives a response from the data owner after the new virtual server requests data from the data owner, it adds the new record to the global key storage with `UUID`. Since the `UUID` is generated by the hypervisor of the physical machine and cannot be faked by a malicious virtual server, it is used for authentication and authorization later when any application running in the virtual server requests data from the data owner, according to the key distribution protocol in Section 5.5.2.

Migration: After the virtual server is migrated from one physical machine to another, the virtual server can continue to access the global key storage from the key client on the new physical machine, since the migration of the virtual server does not change its `UUID`.

Termination: When the data owner terminates a virtual server in the cloud, it notices the key server, which in turn cleans up the global key storage by deleting all data decryption keys associated with the `UUID`.

With support of the above virtual servers' activities in the cloud, the proposed key management framework is able to support key availability across the entire life cycle of a virtual server in a cloud. Besides, the framework achieves the key availability with few overhead to the network which is efficient to support a large-scale dynamic cloud environment.

5.5.4 Discussion

The key management framework is designed to support efficient and secure one-time key usage within a vulnerable virtualization environment in the public cloud. Each time when a virtual server needs to use a cryptographic secrecy, it reads it from a global key storage in stead of from local storage to secure the storage of the one-time cryptographic keys. With this structure, the key management framework is flexible and efficient enough to support a centralized key distribution in a dynamic cloud environment. As the key management framework is built based on a single data center scenario, key distribution across the data center is not in the scoop of this paper.

An alternate solution to support secure key usage is to send the one-time cryptographic secrecy directly from the data owner to the virtual server in the cloud. However, the vulnerable virtual server environment makes the local storage of the secrecy at a risk of leaking. CloudSafe utilizes an isolated secure storage to securely store secrecy and provide an easy access solutions across the entire cloud, which is able to avoid such security issues.

5.6 Implementation & Evaluation

5.6.1 Implementation

In this section, we present our prototype of CloudSafe with Xen virtualzation environment and NFS software including NFS server and NFS client to show the implementation of its main functionality.

Cryptographic primitives. We instantiate our cryptographic definitions with an existing solution proposed in the paper [139]. The solution is a dual-layer algorithm integrated with two existing cryptography: advanced encryption standard (AES) and proxy re-encryption algorithm [28]. The reason that why we choose this algorithm is because the proposed dual-layer algorithm is lightweight and efficient in encryption, re-encryption, and decryption operations, and also because it is comprehensive which includes all the operations we need in our cryptographic scheme. To realize it, we utilize Openssl library [11] for AES functionality in a CFB mode and PBC library from Stanford University [12] for proxy re-encryption algorithm. The code is written by C and be grouped according to each cryptographic primitives as an individual function so that it can be easily used by other application.

Key server. We simplify the implementation of the key server by utilizing network file system (NFS) as shown in Figure 5.6. With the convenient accessibility and scalability of NFS system, the key server is only responsible for communicating with the data owner and storing data decryption keys. While the NFS server is in charge of maintaining the global key storage and connecting with key clients in the cloud. In our prototype, we place both the key server and the NFS server on a dedicated physical machine, manually create and assign *ro* attribute to the folders for individual virtual servers we have in the cluster. We treat the key server as a TCP packet listener, which listens on certain ports and accepts data from the data owner and writes the data through the NFS server.

Key client. We utilize the NFS client to realize the functionality of the key client as shown in Figure 5.7. To obtain the key for the virtual server, the key client firstly read the keys from the global key storage through the NFS client and then passes them to applications in the virtual server

Figure 5.6: The implementation of the key server with NFS.

Figure 5.7: The implementation of a key client with NFS client.

through the in-cloud network. The application running in a virtual server in DomU communicates to the key client via a shared library.

5.6.2 Evaulation

To confirm the practicability of CloudSafe, we demonstrate that CloudSafe brings acceptable cost to the system and the network by answering the following questions:

1. Do we have an efficient proxy re-encryption operations on limited dedicated servers to support one time key strategy?

2. Does the proposed key management framework cause a large network overhead?

Figure 5.8: Comparative results of proxy re-encryption operations on different sizes of encrypted files.

We deployed the prototype of CloudSafe over a computing cluster with 324 individual computers, each of which is Apple Mac Pro with eight Intel Xeon CPU 2.80GHz and 8GB memory. We install Xen virtualization environment on six machines and connect them with Ethernet. A cluster-wide NFS file system is installed and a 10TB storage is mounted to the six machines in a way that the storage is only accessible in Dom0 of these machines.

Re-encryption Efficiency.

To validate the practicality of the strategy of combining one time data decryption key and proxy re-encryption, we focus on the overhead caused by re-encryption operations which is executed by the proxy in CloudSafe. Since one time key strategy requires the data to be re-encrypted each time when the data is accessed, an efficient re-encryption operation is crucial to support the spread of such strategy in the cloud. We testify our algorithm on a single machine equipped with Intel 2 Duo CPU 2.93GHZ, 4GB memory, and Ubuntu 12.4 operating system. We run the code six times on different sizes of encrypted data and average the computation time.

Figure 5.8 shows the computational overhead of the re-encryption operations with different sizes of encrypted data. We can see that the cost of re-encryption operation is independent from the file sizes. The average processing time of re-encryption operation per file is around 0.005 seconds which means 200 data requests can be handled within one seconds. It might be true that the processing time in real application would be a bit larger than the experimental results. We do believe that it will not significantly affect our strategy's practicability in the public cloud environment.

Network Latency

We investigate the network latency caused by the proposed two-phase key distribution protocol in our cluster environment. Recall that CloudSafe utilizes a key server, a NFS server, NFS clients, and key clients to distribute data decryption keys from data owners to virtual servers. Hence, we calculate the network latency with the following equation, where the overall distribution latency of a single data decryption key consists of three parts: the latency between the data owner to the key server, the latency between the key server to a key client (NFS read), and the latency between the key client to the virtual server.

$$Latency_{overall} = Latency_{Dataowner \rightarrow Keyserver} + Latency_{NFSread} + Latency_{Keyclient \rightarrow Virtualserver} \tag{5.1}$$

For latency between the key server to the key client, it takes the fixed 0.003 seconds to distribute a data decryption key which is less than 2048 bits. For the other two kinds of latency, we place

Figure 5.9: Network latency of different key distribution scenarios.

the data owner, the key server, the key client on different machines in the cluster and record the round-trip time of sending different sizes of keys from the data owner to the virtual server. The network setup for the virtual server is bridged which means the virtual server is at the same local area network as the other three components. For comparative purpose, we also record the round-trip latency of direct key distribution from the data owner to the virtual server through the network. We run each experiment six times and average the round-trip latency.

As shown in the Figure 5.9, it is obvious that CloudSafe brings extra network latency, which grows along with the increasing of the size of the key. Note that the extra network latency caused by CloudSafe is exaggerated in our experiment because the network latency of *Direct Key Distribution* is much smaller than real case due to the fact the data owner and the virtual server are located in the same LAN in our experiment.

We note that the two phase key distribution of CloudSafe is not necessarily consecutive. When a virtual server needs to use the data decryption key, the latency for key distribution is essentially caused by the NFS read and phase two key distribution as long as phase one key distribution is

finished in advance. We evaluate phase two key distribution through the network latency from Dom0 and DomU on a single machine. As shown in the Figure 5.9, it takes less than 0.001 seconds to deliver a 2048 bits key. With fixed NFS read operation, the runtime overhead caused by CloudSafe for key distribution should be at most 0.004 seconds. Given this small overhead, we believe that CloudSafe brings acceptable network latency to the real usage.

5.7 Chapter Summary

We focused on the additional security problem caused by malicious cloud customers in the cloud virtualization environment. We proposed CloudSafe, a framework that provides systematic protection of data accessing and processing in vulnerable public cloud environment. Consider that side-channel attack is a long-term and critical threat to cloud customers in public cloud, CloudSafe focuses on reducing the attack surface of variant side-channel attacks by using one-time data decryption keys in cloud. To support frequent data access and key updating, CloudSafe leverages a dedicated in-cloud proxy and the key distribution framework to achieve security and performance requirements. Our evaluation with an implemented prototype confirms that CloudSafe is a practical solution to support large-scale cloud applications.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

In this dissertation, we addressed two security issues of outsourcing data services with the public cloud. We investigated the challenges behind those issues and proposed service-centric security solutions to provide cryptographically guaranteed outsourcing data protection and systematically enforced secure data service in the cloud. For cryptographically guaranteed outsourcing data protection, we designed a proxy-based dual cryptographic scheme to provide secure and efficient data outsourcing service in the public cloud environment. Our scheme leverage two advanced cryptographic algorithms including symmetric encryption and proxy re-encryption. Although our cryptographic scheme is derived from a specific proxy re-encryption algorithms, our scheme is different from it in terms of practicality and efficiency on large-scale data application. Particu-

larly, we proposed two enhancement for proxy re-encryption algorithms. Firstly, we improved the efficiency of all cryptographic operations needed in the previous proxy re-encryption and made them computational comparable to standard AES algorithms; Secondly, we applied re-encryption operation directly on the encrypted data, instead of data decryption keys, to enable efficient and flexible user revocation. Our scheme was experimentally practical to for secure outsourcing data in the public cloud.

Then, we applied our enhanced proxy-based cryptographic scheme to two specific cloud services: cloud-based content delivery service and cloud-based data processing service. By doing this, we aimed to develop service-specific security solutions to enhance practicality of current data-centric security solutions for the public cloud services. Hence, besides security issues of outsourcing data in the public loud, we handled unique challenges in these two specific public cloud services. For cloud-based content delivery service, we developed a novel content delivery model to preserve the efficiency in the delivery network by considering cache property of content delivery network. For cloud-based data processing service, we developed a centralized key management framework to achieve secure key usage by addressing cross-VM side channel attack by malicious cloud customers in this service. Through these application, we strive to demonstrate that relying on merely cryptographic solutions to securely outsourced data service with untrusted public cloud is not enough. The comprehensive solution with cryptographic techniques and systematic mechanisms is the desired direction for future cloud security research.

Beside cloud security research presented in this dissertation [139, 140], I have also explored two other security research areas during my Ph.D study. One area is about malware detection. In

this area, we studied the causal relationship between user's activities and outbound network traffic on a single host machine and proposed a host-based security tools to identify suspicious network traffic [138, 142]. The other area is about continuous authentication. In this area, we proposed a novel authentication methods by developing activity-based personal questions to enhance the security of question-based authentication systems. These questions were generated from a user's network activities, physical events, and conceptual opinions [30].

6.2 Future Work

In our research, we have presented our service-centric solutions for specific cloud services. There are still a wide range of future research opportunities on improving cloud security. We discuss them in the remaining part of this section.

6.2.1 Secure Cloud Collaboration Services

Sharing and co-editing data through public cloud platform is a newly cloud service. Such service allows users to upload data to a center cloud storage and then to share and collaborate with others who have access to the data. Each authorized user are both data contributor and data consumer. However, these convenient functionality introduces extra security requirements to the current cloud-based data sharing scheme. On one hand, data encryption algorithms need to be more efficient. As it is still difficult to compute directly on encrypted data, a user has to decrypt the data first and edit on the plaintext in the cloud. To perform such cryptographic operations with

real-time cloud collaboration services, we need to develop more efficient cryptographic tools. On the other hand, access control mechanisms over outsourced data needs to be more flexible. As the data is changing by authorized data contributors during the runtime of cloud collaboration services, how to control dynamic data remains a problem that needs to be solved in the future. Our current work dose not consider multi-contributor and multi-consumer schemes in cloud-based data sharing services but may have potential to solve the mentioned problems. This is because that the proxy re-encryption algorithm we use in our scheme is able to efficiently transform encrypted data from different origins to a single master copy and vice versa, which is suitable for multi-contributors to edit single copy data in the cloud. Hence, in the future, it is promising to expand our solution to support secure collaboration in the cloud.

6.2.2 Secure Live Migration in the Cloud

In section 2.1.1, we discussed newly developed threats towards cloud infrastructure, such as attacks against virtual machines and hypervisors. Recently, new vulnerabilities of virtual machine (VM) live migration in the cloud [96, 117] pose additional security risk. Security mechanisms, such as source authentication and migration confidentiality and integrity, are needed to support secure live migration in the cloud. However, to our best knowledge, security researches on this topic is at its early stage and no efficient solutions have been proposed. In this dissertation, we assume the data is secure when it is storing in the virtual machine. However, insecure live migration of virtual machines in the cloud throws risks on it. To apply our solution in real system, we need to remove this assumption and explore extra solutions to protect the data via live migration of the virtual

machine over the public cloud infrastructures. One direction to provide such security protection is to utilize the propose key management framework in building secure connection between physical machines during live virtual machine migration. More researches are needed in the future.

Bibliography

[1] Amazon EC2 Dedicated Instances. http://aws.amazon.com/dedicated-instances/.

[2] Amazon security. how to share and use public amis in a secure manner, june 2011. http://aws.amazon.com/articles/0155828273219400.

[3] Amazon Web Services. http://aws.amazon.com.

[4] boto: Python interface to amazon web services. http://code.google.com/p/boto/.

[5] Gogrid dedicated servers, http://www.gogrid.com/products/infrastructure-dedicated-servers.

[6] How secure is dropbox. http://www.dropbox.com/help/27.

[7] IBM Virtual Trust Platform Module. http://researcher.watson.ibm.com/researcher/view_project.php?id=2850.

[8] Ipsec. http://en.wikipedia.org/wiki/IPsec.

[9] Microsoft cloud solutions. http://www.microsoft.com/en-us/cloud/default.aspx?fbid=X9mg968d197\#tab5-small.

[10] Netflix on Amazon's Cloud. http://www.techflash.com/seattle/2010/05/netflix_on_amazon_cloud.html.

[11] OpenSSL Cryptography and SSL/TLS Tookit, http://www.openssl.org/.

[12] Pairing-based cryptography (pbc) library, http://crypto.stanford.edu/pbc/.

[13] The secure shell (ssh) transport layer protocol. http://tools.ietf.org/html/rfc4253.

[14] The transport layer security (tls) protocol version 1.2. http://tools.ietf.org/html/rfc5246.

[15] Xen. http://xen.org/.

[16] Xen: Vulnerabilities Statistics. http://www.cvedetails.com/vendor/6276/XEN.html.

[17] Divertible Protocols and Atomic Proxy Cryptography. In *Advances in Cryptology - EUROCRYPT '98, International Conference on the Theory and Application of Cryptographic Techniques, Espoo, Finland, May 31 - June 4, 1998, Proceeding*, pages 127–144, 1998.

[18] Cloud Computing, an IDC update, 2010.

[19] AWS Customer Agreement http://aws.amazon.com/agreement/, 2011.

[20] O. Aciicmez and c. K. Koç. Trace-driven cache attacks on aes (short paper). In *Proceedings of the 8th international conference on Information and Communications Security*, ICICS'06, pages 112–121, Berlin, Heidelberg, 2006. Springer-Verlag.

[21] A. Adya, W. J. Bolosky, M. Castro, G. Cermak, R. Chaiken, J. R. Douceur, J. Howell, J. R. Lorch, M. Theimer, and R. P. Wattenhofer. Farsite: Federated, Available, and Reliable Storage for An Incompletely Trusted Environment. In *Proceedings of the 5th symposium on Operating systems design and implementation*, OSDI '02, pages 1–14, New York, NY, USA, 2002. ACM.

[22] M. Al-Ibrahim and J. Pieprzyk. Authenticating multicast streams in lossy channels using threshold techniques. In *Proceedings of the First International Conference on Networking-Part 2*, ICN '01, pages 239–249, London, UK, UK, 2001. Springer-Verlag.

[23] A. S. Anne Canteaut, Cedric Lauradoux. Understanding Cache Attacks. Technical Report RR-5881, INRIA, April 2006.

[24] M. J. Atallah, F. Kerschbaum, and W. Du. Secure and Private Sequence Comparisons. In *Proceedings of the 2003 ACM workshop on Privacy in the electronic society*, WPES '03, pages 39–44, New York, NY, USA, 2003. ACM.

[25] M. J. Atallah and J. Li. Secure Outsourcing of Sequence Comparisons. *Int. J. Inf. Secur.*, 4(4):277–287, Oct. 2005.

[26] M. J. Atallah, K. N. Pantazopoulos, J. R. Rice, and E. H. Spafford. Secure Outsourcing of Scientific Computations. *Advances in Computers*, 54:215–272, 2001.

[27] G. Ateniese, R. Di Pietro, L. V. Mancini, and G. Tsudik. Scalable and Efficient Provable Data Possession. In *Proceedings of the 4th international conference on Security and privacy in communication netowrks*, SecureComm '08, pages 9:1–9:10, New York, NY, USA, 2008. ACM.

[28] G. Ateniese, K. Fu, M. Green, and S. Hohenberger. Improved Proxy Re-encryption Schemes with Applications to Secure Distributed Storage. *ACM Trans. Inf. Syst. Secur.*, 9:1–30, February 2006.

[29] A. Aviram, S. Hu, B. Ford, and R. Gummadi. Determinating timing channels in compute clouds. In *Proceedings of the 2010 ACM workshop on Cloud computing security workshop*, CCSW '10, pages 103–108, New York, NY, USA, 2010. ACM.

[30] A. Babic, H. Xiong, D. Yao, and L. Iftode. Building robust authentication systems with activity-based personal questions. In *Proceedings of the 2nd ACM workshop on Assurable and usable security configuration*, SafeConfig 09, pages 19–24, New York, NY, USA, 2009. ACM.

[31] M. Balduzzi, J. Zaddach, D. Balzarotti, E. Kirda, and S. Loureiro. A security analysis of amazon's elastic compute cloud service. In *SAC@SAC 2012, 11th edition of the Computer Security track at the 27th ACM Symposium on Applied Computing, March 26-30, 2012, Trento, Italy*, Trento, ITALY, 03 2012.

[32] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield. Xen and the Art of Virtualization. *SIGOPS Oper. Syst. Rev.*, 37(5):164–177,

Oct. 2003.

[33] P. Bégiun and J.-J. Quisquater. Fast Server-Aided RSA Signatures Secure Against Active Attacks. In *Proceedings of the 15th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '95, pages 57–69, London, UK, UK, 1995. Springer-Verlag.

[34] D. Benjamin and M. Atallah. Private and Cheating-Free Outsourcing of Algebraic Computations. In *Privacy, Security and Trust, 2008. PST '08. Sixth Annual Conference on*, pages 240 –245, oct. 2008.

[35] K. Benson, R. Dowsley, and H. Shacham. Do You Know Where Your Cloud Files Are? In *Proceedings of the 3rd ACM workshop on Cloud computing security workshop*, CCSW '11, pages 73–82, 2011.

[36] D. J. Bernstein. Cache-timing attacks on aes. Technical report, 2005.

[37] G. Bertoni, V. Zaccaria, L. Breveglieri, M. Monchiero, and G. Palermo. Aes power attack based on induced cache miss and countermeasure. In *Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC'05) - Volume I - Volume 01*, ITCC '05, pages 586–591, Washington, DC, USA, 2005. IEEE Computer Society.

[38] M. Blaze. A Cryptographic File System for UNIX. In *Proceedings of the 1st ACM conference on Computer and communications security*, CCS '93, pages 9–16, New York, NY, USA, 1993. ACM.

[39] D. Boneh and X. Boyen. Efficient Selective-ID Secure Identity Based Encryption Without Random Oracles. In *Advances in Cryptology—EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 223–238. Berlin: Springer-Verlag, 2004. Available at `http://www.cs.stanford.edu/~xb/eurocrypt04b/`.

[40] D. Boneh, G. Durfee, and M. K. Franklin. Lower bounds for multicast message authentication. In *Proceedings of the International Conference on the Theory and Application of Cryptographic Techniques: Advances in Cryptology*, EUROCRYPT '01, pages 437–452, London, UK, UK, 2001. Springer-Verlag.

[41] D. Boneh and M. K. Franklin. Identity-based Encryption from the Weil Pairing. In *CRYPTO '01*, London, UK, 2001.

[42] D. Boneh and M. K. Franklin. Identity-based encryption from the weil pairing. In *Proceedings of the 21st Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '01, pages 213–229, London, UK, UK, 2001. Springer-Verlag.

[43] D. Boneh, B. Lynn, and H. Shacham. Short Signatures from the Weil Pairing. In *Proceedings of ASIACRYPT '01*, London, UK, 2001.

[44] B. Briscoe. MARKS: Multicast Key Management using Arbitratily Revealed Key Sequences. In *Proceedings of NGC'99*, 1999.

[45] B. Briscoe. Nark: Receiver-based Multicast Non-repudiation and Key Management. In *Proceedings of EC'99*, 1999.

[46] S. Bugiel, S. Nürnberger, T. Pöppelmann, A.-R. Sadeghi, and T. Schneider. Amazonia: When Elasticity Snaps Back. In *Proceedings of the 18th ACM conference on Computer and communications security*, CCS '11, pages 389–400, New York, NY, USA, 2011. ACM.

[47] S. Butt, H. A. Lagar-Cavilla, A. Srivastava, and V. Ganapathy. Self-service cloud computing. In *Proceedings of the 2012 ACM conference on Computer and communications security*, CCS '12, pages 253–264, 2012.

[48] R. Canetti, J. Garay, G. Itkis, D. Micciancio, M. Naor, and B. Pinkas. Multicast Security: A Taxonomy and Some Efficient Constructions. In *INFOCOM '99.*, March 1999.

[49] R. Canetti and S. Hohenberger. Chosen-ciphertext Secure Proxy Re-encryption. In *Proceedings of the 14th ACM conference on Computer and communications security*, CCS '07, pages 185–194, New York, NY, USA, 2007. ACM.

[50] N. Cao, C. Wang, M. Li, K. Ren, and W. Lou. Privacy-Preserving Multi-Keyword Ranked Search over Encrypted Cloud Data. In *INFOCOM, 2011 Proceedings IEEE*, pages 829 –837, april 2011.

[51] G. Cattaneo, L. Catuogno, A. D. Sorbo, and P. Persiano. The design and implementation of a transparent cryptographic file system for unix. In *Proceedings of the FREENIX Track: 2001 USENIX Annual Technical Conference*, pages 199–212, Berkeley, CA, USA, 2001. USENIX Association.

[52] H. Chen, X. Ma, W. Hsu, N. Li, and Q. Wang. Access Control Friendly Query Verification for Outsourced Data Publishing. In *Proceedings of the 13th European Symposium on Research in Computer Security: Computer Security*, ESORICS '08, pages 177–191, 2008.

[53] Y.-P. Chiu, C.-L. Lei, and C.-Y. Huang. Secure Multicast Using Proxy Encryption. In *Information and Communications Security*, Lecture Notes in Computer Science. 2005.

[54] P. Colp, M. Nanavati, J. Zhu, W. Aiello, G. Coker, T. Deegan, P. Loscocco, and A. Warfield. Breaking up is hard to do: security and functionality in a commodity hypervisor. In *Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles*, SOSP '11, pages 189–202, New York, NY, USA, 2011. ACM.

[55] S. De Capitani di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, G. Pelosi, and P. Samarati. Preserving Confidentiality of Security Policies in Data Outsourcing. In *Proceedings of the 7th ACM workshop on Privacy in the electronic society*, WPES '08, pages 75–84, 2008.

[56] S. De Capitani di Vimercati, S. Foresti, S. Paraboschi, G. Pelosi, and P. Samarati. Efficient and Private Access to Outsourced Data. In *Distributed Computing Systems (ICDCS), 2011 31st International Conference on*, pages 710 –719, june 2011.

[57] S. D. C. di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, and P. Samarati. A Data Outsourcing Architecture Combining Cryptography and Access Control. In *Proceedings of the 2007 ACM workshop on Computer security architecture*, CSAW '07, pages 63–69, New York, NY, USA, 2007. ACM.

[58] S. D. C. di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, and P. Samarati. Over-Encryption: Management of Access Control Evolution on Outsourced Data. In *Proceedings of the 33rd international conference on Very large data bases*, VLDB '07, pages 123–134, 2007.

[59] S. Evdokimov and O. Günther. Encryption Techniques for Secure Database Outsourcing. In *ESORICS*, pages 327–342, 2007.

[60] B. Ford. Icebergs in the clouds: the other risks of cloud computing. *CoRR*, abs/1203.1979, 2012.

[61] B. Ford. Plugging side-channel leaks with timing information flow control. In *Proceedings of the 4th USENIX conference on Hot Topics in Cloud Ccomputing*, HotCloud'12, pages 24–24, 2012.

[62] K. Fu, M. F. Kaashoek, and D. Mazières. Fast and secure distributed read-only file system. *ACM Trans. Comput. Syst.*, 20(1):1–24, Feb. 2002.

[63] V. Goyal, O. Pandey, A. Sahai, and B. Waters. Attribute-based Encryption for Fine-grained Access Control of Encrypted Data. In *Proc. of ACM CCS*, 2006.

[64] M. Green and G. Ateniese. Identity-Based Proxy Re-encryption. In *Applied Cryptography and Network Security*, volume 4521 of *Lecture Notes in Computer Science*, pages 288–306. 2007.

[65] L. Gu, A. Vaynberg, B. Ford, Z. Shao, and D. Costanzo. Certikos: a certified kernel for secure cloud computing. In *Proceedings of the Second Asia-Pacific Workshop on Systems*, APSys '11, pages 3:1–3:5, New York, NY, USA, 2011. ACM.

[66] H. Harney and E. Harder. Logical key hierarchy protocol, 1999.

[67] H. Harney and C. Muckenhirn. Group key management protocol (gkmp) architecture, 1997.

[68] H. Harney and C. Muckenhirn. Group key management protocol (gkmp) specification, 1997.

[69] I. H. Harney and E. Harder. Group secure association key management protocol, 1999.

[70] I. H. Harney and E. Harder. Multicast security management protocol (msmp) requirements and policy, 1999.

[71] C. Inc. Cisco Visual Networking Index: Forecast and Methodology, 2010-2015. White paper, Cisco., 2011.

[72] A.-A. Ivan and Y. Dodis. Proxy Cryptography Revisited. In *NDSS*, 2003.

[73] M. Jakobsson. On Quorum Controlled Asymmetric Proxy Re-encryption. In *Public Key Cryptography*, volume 1560 of *Lecture Notes in Computer Science*, pages 632–632. Springer Berlin / Heidelberg, 1999. 10.1007/3-540-49162-7_9.

[74] E. jin Goh, H. Shacham, N. Modadugu, and D. Boneh. Sirius: Securing Remote Untrusted Storage. In *in Proc. Network and Distributed Systems Security (NDSS) Symposium 2003*, pages 131–145, 2003.

[75] B. W. John Bethencourt, Amit Sahai. Ciphertext-Policy Attribute-Based Encryption. In *Proceedings of S&P 2007*, 2007.

[76] M. Kallahalla, E. Riedel, R. Swaminathan, Q. Wang, and K. Fu. Plutus: Scalable Secure File Sharing on Untrusted Storage. In *Proceedings of the 2nd USENIX Conference on File and Storage Technologies*, pages 29–42, Berkeley, CA, USA, 2003. USENIX Association.

[77] S. Kamara and K. Lauter. Cryptographic Cloud Storage. In *Financial Cryptography and Data Security*, volume 6054 of *Lecture Notes in Computer Science*, pages 136–149. Springer Berlin / Heidelberg, 2010.

[78] G. O. Karame, S. Capkun, and U. Maurer. Privacy-Preserving Outsourcing of Brute-force Key Searches. In *Proceedings of the 3rd ACM workshop on Cloud computing security workshop*, CCSW '11, pages 101–112, New York, NY, USA, 2011. ACM.

[79] S. Kawamura and A. Shimbo. Fast Server-Aided Secret Computation Protocols for Modular Exponentiation. *Selected Areas in Communications, IEEE Journal on*, 11(5):778 –784, jun 1993.

[80] J. Kelsey, B. Schneier, D. Wagner, and C. Hall. Side channel cryptanalysis of product ciphers. *J. Comput. Secur.*, 8(2,3):141–158, Aug. 2000.

[81] F. Koeune, F. Koeune, J.-J. Quisquater, and J. jacques Quisquater. A timing attack against rijndael. Technical report, 1999.

[82] Y. Koglin, D. Yao, and E. Bertino. Secure Content Distribution by Parallel Processing from Cooperative Intermediaries. *IEEE Transactions on Parallel and Distributed Systems*, 2008.

[83] C. A. Lauradoux. Collision attacks on processors with cache and countermeasures, 2005.

[84] J. Levasseur, V. Uhlig, J. Stoess, and S. Götz. Unmodified Device Driver Reuse and Improved System Dependability via Virtual Machines. pages 17–30.

[85] J. Li, M. Krohn, D. Mazières, and D. Shasha. Secure Untrusted Data Repository (SUNDR). In *Proceedings of the 6th conference on Symposium on Opearting Systems Design & Implementation - Volume 6*, pages 9–9, Berkeley, CA, USA, 2004. USENIX Association.

[86] M. Li, S. Yu, N. Cao, and W. Lou. Authorized Private Keyword Search over Encrypted Personal Health Records in Cloud Computing. In *Proceedings of ICDCS 2011*, 2011.

[87] C. H. Lim and P. J. Lee. Security and Performance of Server-Aided RSA Computation Protocols. In *Proceedings of the 15th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO 95, pages 70–83, London, UK, UK, 1995. Springer-Verlag.

[88] M. C. I. Lockheed Martin, LM Cyber Security Alliance. Awareness, Trust and Security to Shape Government Cloud Adoption. White paper, Cisco, 2010.

[89] P. Maniatis, D. Akhawe, K. Fall, E. Shi, S. McCamant, and D. Song. Do You Know Where Your Data Are?: Secure Data Capsules for Deployable Data Protection. In *Proceedings of the 13th USENIX conference on Hot topics in operating systems*, HotOS'13, pages 22–22, Berkeley, CA, USA, 2011. USENIX Association.

[90] T. Matsuda, R. Nishimaki, and K. Tanaka. Cca proxy re-encryption without bilinear maps in the standard model. In P. Nguyen and D. Pointcheval, editors, *Public Key Cryptography 2010*, volume 6056 of *Lecture Notes in Computer Science*, pages 261–278. Springer Berlin Heidelberg, 2010.

[91] D. Mazieres. Don't Trust Your File Server. In *Proceedings of the Eighth Workshop on Hot Topics in Operating Systems*, pages 113–, Washington, DC, USA, 2001. IEEE Computer Society.

[92] E. Miller, D. Long, W. Freeman, and B. Reed. Strong security for distributed file systems. In *In Proceedings of the 20th IEEE International Performance, Computing, and Communications Conference*, pages 34–40, 2002.

[93] D. G. Murray, G. Milos, and S. Hand. Improving xen security through disaggregation. In *Proceedings of the fourth ACM SIGPLAN/SIGOPS international conference on Virtual execution environments*, VEE '08, pages 151–160, New York, NY, USA, 2008. ACM.

[94] D. Naor, M. Naor, and J. Lotspiech. Revocation and tracing schemes for stateless receivers. In J. Kilian, editor, *Advances in Cryptology, CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 41–62. Springer Berlin Heidelberg, 2001.

[95] M. Naor and B. Pinkas. Efficient Trace and Revoke Schemes. In *Proceedings of the 4th International Conference on Financial Cryptography*, FC '00, London, UK, 2001.

[96] J. Oberheide, E. Cooke, and F. Jahanian. Empirical Exploitation of Live Virtual Machine Migration. In *Proceeding of Black Hat*, march 2008.

[97] A. Oprea and M. K. Reiter. Integrity checking in cryptographic file systems with constant trusted storage. In *Proceedings of 16th USENIX Security Symposium on USENIX Security Symposium*, SS'07, pages 13:1–13:16, Berkeley, CA, USA, 2007. USENIX Association.

[98] D. Page. Theoretical use of cache memory as a cryptanalytic side-channel. Technical Report CSTR-02-003, Department of Computer Science, University of Bristol, June 2002.

[99] D. Page. Defending against cache-based side-channel attacks. *Information Security Technical Report*, 8(1):30 – 44, 2003.

[100] J. M. Park, E. K. P. Chong, and H. J. Siegel. Efficient multicast stream authentication using erasure codes. *ACM Trans. Inf. Syst. Secur.*, 6(2):258–285, May 2003.

[101] C. Percival. Cache Missing for Fun and Profit. Technical report, Ottawa, 2005.

[102] A. Perrig, R. Canetti, D. Song, and J. Tygar. Efficient and secure source authentication for multicast. In *Proceedings of Internet Society Network and Distributed System Security Symposium (NDSS 2001)*, pages 35–46, February 2001.

[103] A. Perrig, R. Canetti, J. D. Tygar, and D. Song. Efficient authentication and signing of multicast streams over lossy channels. In *Security and Privacy, 2000. S P 2000. Proceedings. 2000 IEEE Symposium on*, pages 56–73, 2000.

[104] R. Pletka and C. Cachin. Cryptographic security for a high-performance distributed file system. In *Mass Storage Systems and Technologies, 2007. MSST 2007. 24th IEEE Conference on*, pages 227–232, 2007.

[105] L. T. M. Pomelo. Analysis of Netflix's Security Framework for *Watch Instantly* Service, 2009.

[106] R. A. Popa, J. R. Lorch, D. Molnar, H. J. Wang, and L. Zhuang. Enabling Security in Cloud Storage SLAs with CloudProof. Technical Report MSR-TR-2010-46, Microsoft Research, 2010.

[107] S. Rafaeli and D. Hutchison. A survey of key management for secure group communication. *ACM Comput. Surv.*, 35(3):309–329, Sept. 2003.

[108] T. Ristenpart, E. Tromer, H. Shacham, and S. Savage. Hey, You, Get off of My Cloud: Exploring Information Leakage in Third-party Compute Clouds. In *Proceedings of the 16th ACM conference on Computer and communications security*, CCS '09, pages 199–212, New York, NY, USA, 2009. ACM.

[109] T. Ristenpart, E. Tromer, H. Shacham, and S. Savage. Hey, You, Get Off of My cloud! Exploring Information Leakage in Third-Party Compute Clouds. In *Proceedings of CCS*, 2009.

[110] P. Rohatgi. A compact and fast hybrid signature scheme for multicast packet authentication. In *Proceedings of the 6th ACM conference on Computer and communications security*, CCS '99, pages 93–100, New York, NY, USA, 1999. ACM.

[111] C. Security Alliance. Security Guidance for Critical Areas of Focus in Cloud Computing V2.1, 2009. `https://cloudsecurityalliance.org/csaguide.pdf`.

[112] T. R. Seny Kamara, Charalampos Papamanthou. CS2: A Searchable Cryptographic Cloud Storage System. Technical Report MSR-TR-2011-58, Microsoft Research, 2011.

[113] A. Shamir. How to Share A Secret. *Commun. ACM*, 22, November 1979.

[114] J. Shao. Anonymous ID-Based Proxy Re-Encryption. *Information Security and Privacy*, 7372:364–375, 2012.

[115] J. Shao and Z. Cao. Multi-use Unidirectional Identity-based Proxy Re-encryption from Hierarchical Identity-based Encryption. *Information Sciences*, 206(0):83 – 95, 2012.

[116] A. Sherman and D. McGrew. Key establishment in large dynamic groups using one-way function trees. *Software Engineering, IEEE Transactions on*, 29(5):444–458, 2003.

[117] J. Shetty, A. M. R, and S. G. A Survey on Techniques of Secure Live Migration of Virtual Machine. *International Journal of Computer Applications*, 39(12):34–39, February 2012. Published by Foundation of Computer Science, New York, USA.

[118] E. Shi, J. Bethencourt, T.-H. Chan, D. Song, and A. Perrig. Multi-Dimensional Range Query over Encrypted Data. In *Security and Privacy, 2007. SP '07. IEEE Symposium on*, pages 350 –364, may 2007.

[119] J. Somorovsky, M. Heiderich, M. Jensen, J. Schwenk, N. Gruschka, and L. Lo Iacono. All your clouds are belong to us: security analysis of cloud management interfaces. In *Proceedings of the 3rd ACM workshop on Cloud computing security workshop*, CCSW '11, 2011.

[120] D. Song, E. Shi, I. Fischer, and U. Shankar. Cloud Data Protection for the Masses. *Computer*, January 2012.

[121] E. Stefanov, M. van Dijk, A. Juels, and A. Oprea. Iris: a scalable cloud file system with efficient integrity checks. In *Proceedings of the 28th Annual Computer Security Applications Conference*, ACSAC '12, pages 229–238, New York, NY, USA, 2012. ACM.

[122] C. A. Stein, J. H. Howard, and M. I. Seltzer. Unifying file system protection. In *Proceedings of the General Track: 2002 USENIX Annual Technical Conference*, pages 79–90, Berkeley, CA, USA, 2001. USENIX Association.

[123] U. Steinberg and B. Kauer. Nova: a microhypervisor-based secure virtualization architecture. In *Proceedings of the 5th European conference on Computer systems*, EuroSys '10, pages 209–222, New York, NY, USA, 2010. ACM.

[124] K. Suzaki, K. Iijima, T. Yagi, and C. Artho. Memory Deduplication as A Threat to The Guest OS. In *Proceedings of the Fourth European Workshop on System Security*, EUROSEC '11, pages 1:1–1:6, New York, NY, USA, 2011. ACM.

[125] P. Traynor, K. R. B. Butler, W. Enck, and P. McDaniel. Realizing Massive-Scale Conditional Access Systems Through Attribute-Based Cryptosystems. In *NDSS*, 2008.

[126] E. Tromer, D. A. Osvik, and A. Shamir. Efficient cache attacks on aes, and countermeasures. *J. Cryptol.*, 23(2):37–71, Jan. 2010.

[127] Y. Tsunoo, T. Saito, T. Suzaki, and M. Shigeri. Cryptanalysis of des implemented on computers with cache. In *Proc. of CHES 2003, Springer LNCS*, pages 62–76. Springer-Verlag, 2003.

[128] Y. Tsunoo, E. Tsujihara, M. Shigeri, H. Kubo, and K. Minematsu. Improving cache attacks by considering cipher structure. *Int. J. Inf. Secur.*, 5(3):166–176, July 2006.

[129] C. Wang, N. Cao, J. Li, K. Ren, and W. Lou. Secure Ranked Keyword Search over Encrypted Cloud Data. In *Proceedings of the 2010 IEEE 30th International Conference on Distributed Computing Systems*, ICDCS '10, pages 253–262, Washington, DC, USA, 2010. IEEE Computer Society.

[130] C. Wang, Q. Wang, K. Ren, N. Cao, and W. Lou. Toward secure and dependable storage services in cloud computing. *IEEE Trans. Serv. Comput.*, 5(2):220–232, Jan. 2012.

[131] C. Wang, Q. Wang, K. Ren, and W. Lou. Privacy-Preserving Public Auditing for Data Storage Security in Cloud Computing. In *INFOCOM, 2010 Proceedings IEEE*, pages 1 –9, march 2010.

[132] 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, pages 355–370, 2009.

[133] W. Wang, Z. Li, R. Owens, and B. Bhargava. Secure and Efficient Access to Outsourced Data. In *Proceedings of the 2009 ACM workshop on Cloud computing security*, CCSW '09, pages 55–66, 2009.

[134] B. Waters. Efficient identity-based encryption without random oracles. In *Proceedings of the 24th annual international conference on Theory and Applications of Cryptographic Techniques*, EUROCRYPT'05, pages 114–127, Berlin, Heidelberg, 2005. Springer-Verlag.

[135] C. K. Wong, M. Gouda, and S. S. Lam. Secure Group Communications Using Key Graphs. *IEEE/ACM Trans. Netw.*, 8, February 2000.

[136] Y. Wu, D. Ma, and C. Xu. Efficient object-based stream authentication. In *Proceedings of the Third International Conference on Cryptology: Progress in Cryptology*, INDOCRYPT '02, pages 354–367, London, UK, UK, 2002. Springer-Verlag.

[137] M. Xie, H. Wang, J. Yin, and X. Meng. Integrity Auditing of Outsourced Data. In *Proceedings of the 33rd international conference on Very large data bases*, VLDB '07, pages 782–793, 2007.

[138] H. Xiong, P. Malhotra, D. Stefan, C. Wu, and D. Yao. User-assisted host-based detection of outbound malware traffic. In *Proceedings of the 11th international conference on Information and Communications Security*, ICICS 09, pages 293–307, Berlin, Heidelberg, 2009. Springer-Verlag.

[139] H. Xiong, X. Zhang, D. Yao, X. Wu, and Y. Wen. Towards end-to-end secure content storage and delivery with public cloud. In *Proceedings of the second ACM conference on Data and Application Security and Privacy*, CODASPY '12, pages 257–266, New York, NY, USA, 2012. ACM.

[140] H. Xiong, X. Zhang, W. Zhu, and D. Yao. Cloudseal: End-to-end content protection in cloud-based storage and delivery services. In M. Rajarajan, F. Piper, H. Wang, and G. Kesidis, editors, *Security and Privacy in Communication Networks*, volume 96 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, pages 491–500. Springer Berlin Heidelberg, 2012.

[141] L. Xiong, S. Chitti, and L. Liu. Preserving Data Privacy in Outsourcing Data Aggregation Services. *ACM Trans. Internet Technol.*, 7(3), Aug. 2007.

[142] K. Xu, H. Xiong, C. Wu, D. Stefan, and D. Yao. Data-provenance verification for secure hosts. *IEEE Trans. Dependable Secur. Comput.*, 9(2):173–183, Mar. 2012.

[143] Y. Yang, L. Gu, and F. Bao. Addressing Leakage of Re-encryption Key in Proxy Re-encryption Using Trusted Computing. *Trusted Systems*, 6802:189–199, 2011.

[144] D. Yao, Y. Koglin, E. Bertino, and R. Tamassia. Decentralized Authorization and Data Security in Web Content Delivery. In *Proc ACM Symp. on Applied Computing (SAC)*, 2007.

[145] M. L. Yiu, G. Ghinita, C. Jensen, and P. Kalnis. Outsourcing Search Services on Private Spatial Data. In *Data Engineering, 2009. ICDE '09. IEEE 25th International Conference on*, pages 1140 –1143, 29 2009-april 2 2009.

[146] S. Yu, C. Wang, K. Ren, and W. Lou. Achieving Secure, Scalable, and Fine-grained Data Access Control in Cloud Computing. In *INFOCOM, 2010 Proceedings IEEE*, pages 1 –9, march 2010.

[147] S. Yu, C. Wang, K. Ren, and W. Lou. Attribute Based Data Sharing with Attribute Revocation. In *Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security*, ASIACCS '10, pages 261–270, 2010.

[148] K. Zhang, X. Zhou, Y. Chen, X. Wang, and Y. Ruan. Sedic: Privacy-aware Data Intensive Computing on Hybrid Clouds. In *Proceedings of the 18th ACM conference on Computer and communications security*, CCS '11, pages 515–526, 2011.

[149] Y. Zhang, A. Juels, A. Oprea, and M. Reiter. Homealone: Co-residency Detection in the Cloud via Side-Channel Analysis. In *Security and Privacy (SP), 2011 IEEE Symposium on*, pages 313 –328, may 2011.

[150] Y. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart. Cross-vm side channels and their use to extract private keys. In *Proceedings of the 2012 ACM conference on Computer and communications security*, CCS '12, pages 305–316, 2012.

[151] L. Zhou, M. A. Marsh, F. B. Schneider, and A. Redz. Distributed Blinding for Distributed Elgamal Re-Encryption. In *Proceedings of the 25th IEEE International Conference on Distributed Computing Systems*, ICDCS '05, pages 824–824, Washington, DC, USA, 2005. IEEE Computer Society.

[152] W. Zhou, M. Sherr, W. R. Marczak, Z. Zhang, T. Tao, B. T. Loo, and I. Lee. Towards A Data-centric View of Cloud Security. In *Proceedings of the second international workshop on Cloud data management*, CloudDB '10, pages 25–32, New York, NY, USA, 2010. ACM.

[153] S. Zhu, C. Yao, D. Liu, S. Setia, and S. Jajodia. Efficient Security Mechanisms for Overlay Multicast based Content Delivery. *Comput. Commun.*, 30:793–806, February 2007.