J. Parallel Distrib. Comput. 74 (2014) 2872–2883

Contents lists available at ScienceDirect

J. Parallel Distrib. Comput.

journal homepage: www.elsevier.com/locate/jpdc

Shield: A stackable secure storage system for file sharing in public storage

Jiwu Shu *, Zhirong Shen, Wei Xue

Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China
Tsinghua National Laboratory for Information Science and Technology, Beijing 100084, China

HIGHLIGHTS

- We propose a new system architecture for secure file sharing in cloud scenario.
- We implement a stackable secure storage system named Shield.
- A hierarchical key organization is designed for convenient keys management.
- Shield adopts lazy revocation to accelerate the revocation process.
- Shield supports concurrent write access by employing a virtual linked list.

ARTICLE INFO

Article history:
Received 23 September 2013
Received in revised form
8 June 2014
Accepted 10 June 2014
Available online 19 June 2014

Keywords:
Storage system
Cryptographic controls
Keys management
Proxy server
Secure sharing
Permission revocation
Concurrent writes

ABSTRACT

With the increasing amount of personal data stored in public storage, users are losing control of their physical data, putting their data information at risk of theft or being compromised. Traditional secure storage systems either require users to completely trust the storage provider or impose the considerable burden of managing files on file owners; such systems are inapplicable in the practical cloud environment. This paper addresses these challenging problems by proposing a new secure system architecture and implementing a stackable secure storage system named Shield, in which a proxy server is introduced to be in charge of authentication and access control. We propose a new variant of the Merkle Hash Tree to support efficient integrity checking and file content update; further, we have designed a hierarchical key organization to achieve convenient keys management and efficient permission revocation. Shield supports concurrent write access by employing a virtual linked list; it also provides secure file sharing without any modification to the underlying file systems. A series of evaluations over various real benchmarks show that Shield causes about 7%∼13% performance degradation when compared with eCryptfs but provides enhanced security for user's data.

1. Introduction

As a new kind of online storage over network, cloud storage delivers elastic storage service and provides virtually unlimited storage capacity without requiring users to perform complex system configurations or buy expensive storage devices. Owing to its convenience and economy, data owners are willing to concentrate their data to the cloud.

Although cloud storage dramatically improves the efficiency of data management, data owners have to sacrifice physical control of their data by handing it over to the cloud server, which may put the data information at risk of theft or being compromised caused by unauthorized access. Derived from this worry, the research reports on data leaks have increasingly emerged in recent years, causing public concern about the security when their sensitive data are stored in the public storage. The report released by IDC [8] further points out that data security have been treated as a top priority in cloud computing.

Many secure storage systems [15,17,13,40,3,14,16,19] have been proposed for protecting data security by using several key technologies such as encrypt-on-disk [17], but most of them are mostly based on the outdated models of either requiring the cloud server to be completely trusted [28,25] to execute access control and key distribution, or needing file owners to actively manage security themselves [17,26] (i.e. only trusting themselves and deal with users' access requests by themselves). However, the two

* Corresponding author at: Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China.
E-mail addresses: shujw@tsinghua.edu.cn (J. Shu), zhirong.shen2601@gmail.com, czr10@mails.tsinghua.edu.cn (Z. Shen), xuewei@tsinghua.edu.cn (W. Xue).

http://dx.doi.org/10.1016/j.jpdc.2014.06.003
0743-7315/

extreme trust models, complete trust or no trust, either invite potential security threats or involve significant inconveniences. When the cloud server is given complete trust, users may worry about the status of their sensitive data once the cloud server does not behave neutrally or is fully controlled by hackers, and the cloud server may also fear to get into the risks of economic disputes and reputation losses, once user's data are leaked because of either sudden accidents or careless operations. In another case, if file owners are required to bear the burden of trusting themselves, they will be forced to be always online to handle access requests and execute access control, causing a considerable management burden. There are extra works that make some impressive efforts to alleviate the shortcomings of the above two architectures. For example, Castiglione et al. [5] try to move the management burden to users by utilizing secret sharing mechanism. They require any data access should receive the permissions from at least k users. However, it may also introduce considerable computation and management burden to users. Though another work [7] introduces a generic credential authority to manage keys and enforce access control. Users in this architecture are responsible for keys management for the version signature. This may cause considerable management burden once a user joins many groups. Meanwhile, it also introduces many public-key encryption computations, which are far more expensive than symmetric-key encryption.

To address these problems, we analyze the threat models of file sharing among multiple users in public storage, establish a new trust system, and propose a new system architecture in which *users can store files and share them efficiently under multi-party shared public storage and network environments*. Based on this architecture, we develop a stackable secure storage system named Shield, which strives to provide secure file sharing and free file owners from cumbersome keys management in the cloud scenario. To improve portability, Shield requires no modification to the file system and can be directly deployed on top of existing file systems to provide extended end-to-end security and efficient access control, both of which are independent of cloud storage systems or administrators.

In addition, to prevent the cloud server from accessing plaintext when encrypting/decrypting data, Shield migrates operations of data encryption/decryption and integrity checking to be performed at the client side. This change also benefits the scalability of the whole system. Moreover, Shield concentrates on the mechanisms that provide efficient permission revocation and support write concurrency when data files are shared among multiple users. Our contributions are as follows:

First, we propose a new architecture for secure file sharing that neither places complete trust over the cloud server nor imposes cumbersome management burdens on file owners. In this architecture, we use the *proxy server* (PS) to manage access control and distribute secret keys. To avoid the problem of the PS being a bottleneck, a PS-Group can be easily constructed to decrease the burden and trust over every individual PS.

Second, we develop a hierarchical key organization to lighten the complexity of keys management, design a variant of the *Merkle Hash Tree* (MHT) for integrity checking, adopt lazy revocation to improve the efficiency of revocation operations, and exploit the virtual hash linked list to support concurrent writing to a file.

Third, by employing some representative benchmarks, intensive tests are conducted to evaluate the performance of Shield. The final results show that Shield causes about 7%–13% performance degradation when compared with eCryptfs but provides enhanced security and a single PS can support more than 45,000 users' requests in one second.

The remainder of the paper continues as follows: We review the related work in Section 2, describe the design goals and assumptions in Section 3, and introduce the key techniques in Section 4. Section 5 describes the protocols for file reading, file writing, and permission revocation. Subsequently, we analyze the security of Shield in Section 6, discuss its implementation in Section 7, and evaluate its performance in Section 8. Finally, we conclude our work in Section 9.

2. Related work

CFS [3] is the earliest work of encrypt-on-disk file systems, which uses a single key to encrypt the whole file directory. As its variants, Cryptfs [40], Cepheus [32], and TCFS [6] are proposed later. Cryptfs associates symmetric keys and file groups, allowing group file sharing. Cepheus introduces a lockbox for group management and firstly proposes lazy revocation. TCFS designs transparent cryptographic file systems by integrating the encryption service with file systems. However, all of them miss the considerations of read–write differentiation and lack an efficient keys management mechanism.

To support wide file sharing service, NCryptfs [37] implemented at the kernel level supports multi-user sharing on the same machine. The Swallow [30] implements access control by executing encryption at the client level. However, Swallow neither offers file-sharing nor differentiates read–write operations. NASD [14] provides data security for network-attached storage. It keeps data in the form of plaintext and the security guarantee requires the participation of storage devices.

SFS [25] has to rely on the trusted server to enforce access control. It provides authentication for remote file systems, and the communication channel with the server is encrypted by a session key. CryptosFS [28] also trusts the storage servers to verify user's access and uses public-key encryption instead of existing access control mechanism in NFS to regulate user's access. Derived from CryptosFS, eCryptfs [16] is designed to enforce data confidentiality on secondary storage belonging to a single host, however, it cannot support file sharing because of the absence of keys management and access control.

With the system scales up, the data reliability and durability becomes important design criteria for large storage systems. OceanStore [20] and FARSITE [1] mainly focus on the availability and fault-tolerance while providing security for those distributed file systems. To protect long-term data, POTSHARDS [33] uses secret splitting and approximate pointers to secure data, which may only be cracked after decades, and SafeStore [19] spreads data across autonomous SSPs using informed hierarchical erasure coding to increase data durability.

The Secure Untrusted Data Repository (SUNDR) [24] relies on a storage server to execute access control while providing data confidentiality with per-file key encryption and file integrity with hash trees. PCFS [12] is a file system with proof-carrying authorization that provides access control with policy support by formal proof and capability. Maat [21] is designed for object-based storage and it uses extended capabilities, automatic revocation and secure delegation to secure distributed file systems. SNAD [26] employs a lockbox to protect integrity. However, neither Maat nor PCFS can provide on-disk security, leaving data exposed to adversaries. What is more, all of SUNDR [24], SNAD [26] and Maat [21] require new types of storage servers, while Shield does not demand a new infrastructure and allows users to manage their own access without relying on storage servers.

Besides disordering data by encryption, some representative works protect data security by forbidding the illegal access to the data. I3FS [29] is a file system with build-in integrity checking that uses cryptographic checksums to provide integrity validation for files. Kerberos [27] provides authentication service for clients in insecure network environments. Clients have to interact with AS (Authentication Server) and TGS (Ticket Granting Server) for authentication before applying for this service.