B. Mitschang et al. (Hrsg.): Datenbanksysteme für Business, Technologie und Web (BTW 2017),
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2017 625

Secure Cryptographic Deletion in the Swift Object Store

Tim Waizenegger[1]

Abstract: The secure deletion of data is of increasing importance to individuals, corporations as well as governments. Recent data breaches as well as advances in laws and regulations show that secure deletion is becoming a requirement in many areas. However, this requirement is rarely considered in today's cloud storage services. The reason is that the established processes for secure deletion of on-site storage are not applicable to cloud storage services. Cryptographic deletion is a suitable candidate for these services, but a research gap still exists in applying cryptographic deletion to large cloud storage services. For these reasons, we demonstrate a working prototype for a secure-deletion enabled cloud storage service with the following two main contributions: A model for offering high value service without full plain-text access to the provider, as well as secure deletion of data through cryptography.

Keywords: secure data deletion, cryptographic deletion, data erasure, records management, retention management, key management, data shredding

1 Background

Cloud based storage solutions are a popular service today especially among consumers. They are used for synchronizing data across devices, for backup and archiving purposes, and for enabling access at any time from anywhere. But the adoption of such storage services still faces many challenges in the government and enterprise sector. The customers, as well as the vendors, have a desire to move these systems, or parts of these systems, to cloud environments in order to reduce cost and improve the service. But security issues often prevent customers from adopting cloud storage services.

Cloud storage providers address these issues by offering data encryption in various configurations. The main difference in their implementation of data encryption is the management of encryption keys, and especially the authority over master keys. Cloud providers generally prefer an encryption system where they keep the master keys and have access to unencrypted data. This enables the providers to operate on the customers' data in order to offer advanced services like indexing/searching the data or analyzing it. Generally, a provider with access to unencrypted data is able to provide a higher quality and more useful service to the customer. Providers that allow client-side encryption and do not store any master keys do not have access to the data. They can only operate as data-dumps and offer very limited features to the customer, which makes this an unpopular business model.

Another security aspect that is especially important to government and enterprise customers is the secure removal of deleted data, i.e. disposing of data after its lifetime has passed. Recent

[1] University of Stuttgart, Institute for Parallel and Distributed Systems, Applications of Parallel and Distributed Systems, waizentm@ipvs.uni-stuttgart.de

data breaches have shown that consumer data which was assumed to be deleted could be restored by attackers, causing privacy issues [Os15]. Enterprises and especially governments are bound by a myriad of regulations for information life cycle governance [EP16]. These state specific requirements for how contracts, reports, personal information, and others have to be stored, and when and how they have to be deleted.

Individuals, corporations as well as governments have a requirement for secure data deletion for these reasons. Not only because they want to be compliant with the law, but also in order to avoid storing compromising information when they do not have to.

Today, secure deletion is usually done by physically destroying storage media in the enterprise and government sector. Individuals mostly rely on "virtual shredding" (i.e. overwriting storage blocks). In a cloud storage scenario, both methods are no longer applicable. Physical disks are shared among different applications and even customers. Providers use complex tiered-storage setups or outsource the physical storage themselves. Identifying the physical disks that need to be destroyed, or the blocks that need to be overwritten, becomes difficult to impossible [DW10].

For these reasons, we propose to use data encryption in order to provide secure deletion; i.e. apply cryptographic deletion to cloud storage services. We achieve this by using a key-management based on our Key-Cascade approach [Ba16, Wa17]. We further propose the separation of data and metadata in cloud storage services. This provides the opportunity to encrypt the data client-side and only allow the provider access to metadata. This should incentivize more providers to offer client-side encryption because they can still offer advances services on the metadata.

In this demo, we present a cloud storage system with the two main contributions:

1. Separation of data and metadata to allow the provider to access unencrypted metadata for enabling advances services.
2. Data encryption with a key management that enables cryptographic deletion.

2 MCM Functionality and Architecture

In order to benchmark our key-management mechanism and evaluate the data/metadata separation, we built the Micro Content Management system MCM[2] which will be presented in this Demo.

MCM is based on Enterprise Content Management systems like Box[3]. It stores objects and files inside storage containers in the Swift[4] object store. Whole containers can be transparently encrypted with a key-management mechanism that allows secure deletion of individual objects. Our user interface allows uploading and retrieving files, setting retention

[2] https://github.com/timwaizenegger/mcm-sdos
[3] https://www.box.com
[4] http://docs.openstack.org/developer/swift/

dates and scheduling deletion, extracting and viewing metadata, and analyzing and graphing analyses on this metadata. It also features interactive visualizations of the underlying key management data structures.

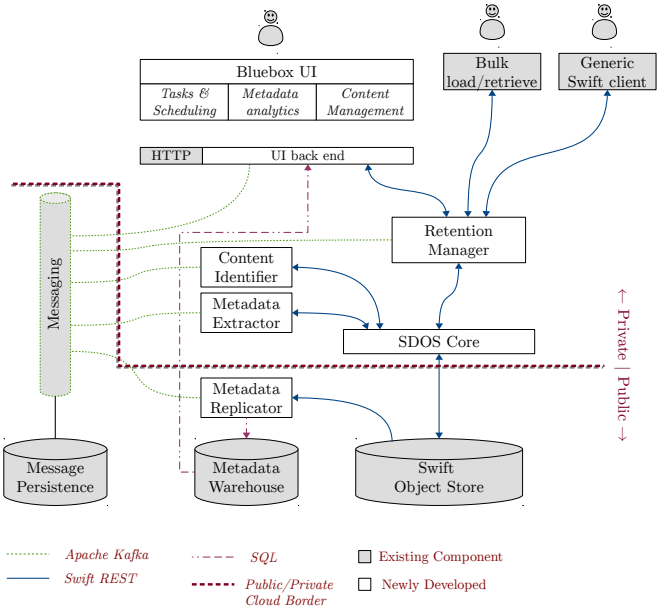

Fig. 1: The Architecture of the Micro Content Management System.

Figure 1 shows the high level architecture of MCM. We use three data management systems (bottom row of Figure 1): An Apache Kafka streaming platform for loosely coupled communication, an SQL database for storing and analyzing the unencrypted metadata, and a Swift object store that holds all the (encrypted) data objects. In order to interface the encryption as well as retention management components with the Swift object store, we designed these services as API proxies for the Swift REST protocol. This enables us to use any unmodified Swift backend (e.g. SaaS) as well as any existing Swift clients. The SDOS encryption and retention manager form a flexible pipeline. All MCM components can run multithreaded or distributed to enable horizontal scaling and high availability. The Kafka streaming platform is used for triggering the execution of jobs for metadata extraction and replication as well as scheduled deletion of old objects. We use a separate metadata warehouse, as Swift lacks advanced querying capabilities for metadata (only retrieving and listing is possible). All the object metadata is primarily stored in Swift and then replicated to the warehouse for analysis. This warehouse is implemented using a relational DBMS because i) the metadata schema is known from the extraction phase and fits well with the relational model, and ii) the intended analytical queries can easily be expressed in SQL.

The location where the components from Figure 1 run is critical to the security of the system. In order to guarantee the secure deletion property, the content of the stored objects must never leave a trusted environment in unencrypted form. The same must be guaranteed for the encryption keys. Our SDOS encryption uses a tree structure for key management of

which only the root key must be kept secure. All other keys are stored encrypted on Swift together with the data objects.

One possible separation of trusted/untrusted environment is given by the red line in Figure 1. It shows that all the data storage system can be outsourced to the public (untrusted) environment, because all sensitive data is encrypted. The metadata replicator only copies data between Swift and the database, so it can run on a public cloud as well. The other components handle sensitive, unencrypted data and the SDOS component has to manage the root key, so these components need to run in a trusted environment. An enterprise cloud gateway appliance could host these sensitive components on premise. Enterprise cloud gateways are already used today for storage outsourcing, they could be equipped with our approach to provide secure deletion as well.

3 Demo Overview

This Demo will show the theory behind cryptographic deletion and its key management mechanism as well as present the Micro Content Management system as an example application for cryptographic deletion. In the demo scenario, we will create new storage containers with and without cryptographic deletion and show data ingestion and retrieval using different client applications. Our Bluebox user interface features interactive visualizations of the key management data structures. We will use the visualization to show how insertions/deletions affect the key management data structure. We will demonstrate the proposed separation of encrypted data and unencrypted metadata by extracting metadata from previously ingested emails, pictures and documents. We then show how the tasks for metadata extraction and replication are triggered, how the metadata warehouse tables look like, and how analyses can be realized.

Screenshots of the application, as well as more details about its capabilities, can be found on our github page: `https://github.com/timwaizenegger/mcm-bluebox`

References

[Ba16] Barney, Jonathan; Lebutsch, David; Mega, Cataldo; Schleipen, Stefan; Waizenegger, Tim: Deletion of content in digital storage systems. US Patent 9,298,951, March 2016.

[DW10] Diesburg, Sarah M.; Wang, An-I Andy: A Survey of Confidential Data Storage and Deletion Methods. ACM Comput. Surv., 43(1):2:1–2:37, December 2010.

[EP16] European Parliament, Council: Regulation (EU) 2016/679: General Data Protection Regulation. Article 17 "Right to Erasure", 2016.

[Os15] Osborne, Charlie: Ashley Madison hack: How much user data did "Paid delete" function obliterate? ZDNet Security, 2015.

[Wa17] Waizenegger, Tim; Lebutsch, David; Mega, Cataldo; Mitschang, Bernhard: Design and Implementation of Efficient Key Management for Secure Cryptographic Data Erasure in Large Cloud-Based Storage Systems. Under review for: EDBT 20th International Conference on Extending Database Technology, 2017.