International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 2 Issue 10, October - 2013

A Review Paper on Cryptonite Technique: A Secure and Data Repository on Public Clouds

Mrs. Anjali V. Almale
P.V.P.I.T., Pune University

Prof. Yogesh B.Gurav
P.V.P.I.T., Pune University

Abstract: We consider the problem of building a secure cloud storage service on top of a public cloud infrastructure where the service provider is not completely trusted by the customer. We describe, at a high level, several architectures that combine recent and non-standard cryptographic primitives in order to achieve our goal. Cloud storage has become immensely popular for maintaining synchronized copies of files and for sharing documents with collaborators. However, there is heightened concern about the security and privacy of Cloud-hosted data due to the shared infrastructure model and an implicit trust in the service providers. We survey the benefits such architecture would provide to both customers and service providers and give an overview of recent advances in cryptography motivated specifically by cloud storage.

***Keywords*:** Cloud data storage; secure data sharing, Data repository, Public cloud.

1. Introduction

Emerging needs of secure data storage and sharing for domains like Smart Power Grids, which deal with sensitive consumer data, require the persistence and avail-ability of Cloud storage but with client-controlled security and encryption, low key management overhead, and minimal performance costs. *Cryptonite* is a secure Cloud storage repository that addresses these requirements using a *StrongBox* model for shared key management. We describe the *Cryptonite* service and desktop client, discuss performance optimizations, and provide an empirical analysis of the improvements. Our experiments shows that *Cryptonite* clients achieve a 40% improvement in file upload bandwidth over plaintext storage using the Azure Storage Client API despite the added security benefits, while our file download performance is 5 times faster than the baseline for files greater than 100MB.

Despite its success, there is heightened concern about the security and privacy of Cloud-hosted data [1] to many eBusiness and eSciences that are considering shifting to Clouds. These arise from reasons: shared Cloud infrastructure for storing data from different organizations, co-location of multi-tenant applications with the storage services, and malicious insiders at Cloud providers getting access to data. These do not presuppose ill-intentions by Cloud providers, who make a best effort (but not guarantees) of data security, but are rather the consequence of the "shared infrastructure managed by third party providers" model of Clouds

Storage services based on public clouds such as Microsoft's Azure storage service and Amazon's S3 provide customers with scalable and dynamic storage. By moving their data to the cloud customers can avoid the costs of building and maintaining a private storage infrastructure, opting instead to pay a service provider as a function of its needs. For most customers, this provides several benefits including availability (i.e., being able to access data from anywhere) and reliability (i.e., not having to worry about backups) at a relatively low cost.

To address the concerns outlined above and increase the adoption of cloud storage, we argue for designing a virtual private storage service based on recently developed cryptographic techniques. Such a service should aim to achieve the best of both worlds by providing the security of a private cloud and the functionality and cost savings of a public cloud. More precisely, such a service should provide (at least):

- Confidentiality: the cloud storage provider does not learn any information about customer data
- Integrity: any unauthorized modification of customer data by the cloud storage provider can be detected by the customer while retaining the main benefits of a public storage service:
- Availability: customer data is accessible from any machine and at all times
- Reliability: customer data is reliably backed up
- Efficient retrieval: data retrieval times are comparable to a public cloud storage service
- Data sharing: customers can share their data with trusted parties.

These arise from reasons: shared Cloud infrastructure for storing data from different organizations, co-location of multi-tenant applications with the storage services, and malicious insiders at Cloud providers getting access to data. These do not presuppose ill-intentions by Cloud providers, who make a best effort (but not guarantees) of data security, but are rather the consequence of the "shared infrastructure

managed by third party providers" model of Clouds[2].

1.1 Motivation

A Cloud-hosted data repository to hold personally identifiable information for scientific and operational needs should enforce this single owner/multiple writers/multiple readers permission on individual data files with low management overhead, while ensuring that plaintext data is not stored in the Cloud.

Cloud infrastructures can be roughly categorized as either private or public. In a private cloud, the infrastructure is managed and owned by the customer and located on-premise (i.e., in the customers region of control). In particular, this means that access to customer data is under its control and is only granted to parties it trusts. In a public cloud the infrastructure is owned and managed by a cloud service provider and is located off-premise (i.e., in the service provider's region of control). This means that customer data is outside its control and could potentially be granted to untrusted parties.

While the benefits of using a public cloud infrastructure are clear, it introduces significant security and privacy risks. In fact, it seems that the biggest hurdle to the adoption of cloud storage (and cloud computing in general) is concern over the confidentiality and integrity of data. While, so far, consumers have been willing to trade privacy for the convenience of software services[3].

2. Architecture of a Cryptographic Storage Service

We now describe, at a high level, a possible architecture for a cryptographic storage service. At its core, the architecture consists of three components: a data processor (DP), that processes data before it is sent to the cloud; a data verifier (DV), that checks whether the data in the cloud has been tampered with; and a token generator (TG), that generates tokens that enable the cloud storage provider to retrieve segments of customer data; and a credential generator that implements an access control policy by issuing credentials to the various parties in the system (these credentials will enable the parties to decrypt encrypted files according to the policy). We describe designs for both consumer and enterprise scenarios.

2.1 An Enterprise Architecture

In the enterprise scenario we consider an enterprise MegaCorp that stores its data in the cloud; a business partner PartnerCorp with whom MegaCorp wants to share data; and a cloud storage provider that stores MegaCorp's data.

To use the service, MegaCorp deploys dedicated machines within its network. Depending on the particular scenario, these dedicated machines will run various core components. Since these components make use of a master secret key, it is important that they be adequately protected and, in particular, that the master key be kept secret from the cloud storage provider and PartnerCorp. If this is too costly in terms of resources or expertise, management of the dedicated machines (or specific components) can alternatively be outsourced to a trusted entity.

In the case of a medium-sized enterprise with enough resources and expertise, the dedicated machines include a data processor, a data verifier, a token generator and a credential generator

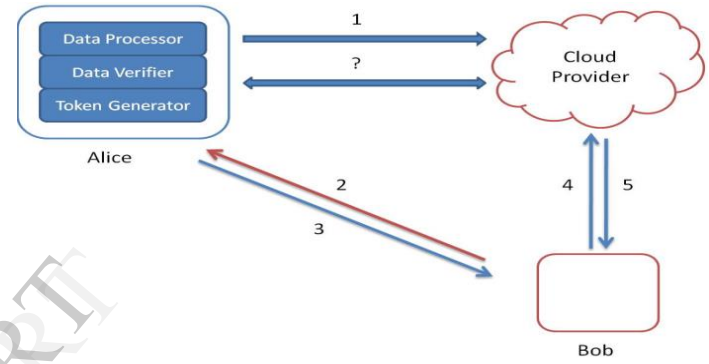


Figure 1: (1) Alice's data processor prepares the data before sending it to the cloud; (2) Bob asks Alice for permission to search for a keyword; (3) Alice's token and credential generators send a token for the keyword and a credential back to Bob; (4) Bob sends the token to the cloud; (5) the cloud uses the token to find the appropriate encrypted documents and returns them to Bob. (?) At any point in time, Alice's data verifier can verify the integrity of the data.

3. About Cryptonite

In this paper, we present *Cryptonite*, a secure Cloud storage respository that addresses these requirements. Our design introduced earlier [5] is validated here, to wit we present an implementation of the *Cryptonite* service and desktop client on the Microsoft Windows Azure Cloud platform. We also discuss optimizations to overcome data security overheads and provide an empirical analysis of the improvements. Specifically, our contributions are as follows

- We implement *Cryptonite*, a secure data repository that runs within Azure and provides service APIs compatible with existing Cloud storage services,
- We present pipelined and data parallel performance optimizations to reduce the security overhead caused through encryption and key management, and
- We experimentally evaluate *Cryptonite* on the Azure Cloud, compare it with baseline client storage access,

and demonstrate the efficacy of our optimizations.

The rest of the paper is organized as follows. In Section we review the *Cryptonite* design. after describes the service and client implementations in next section Related work is given.We present our conclusions in Section 8. Characterize security and privacy **requirements** for data storage and sharing, using the smart power grid domain as motivation.

Cryptonite: An integrated system designed for a shared, secure Data Repository on Cloud.

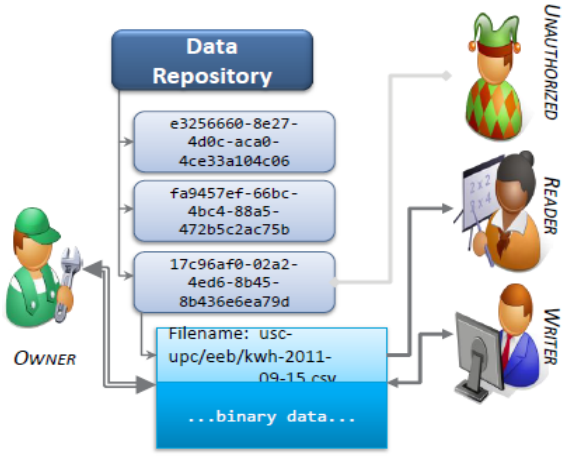
3.1 BACKGROUND: *Cryptonite* DESIGN

The basic tenets behind the *Cryptonite* design are to [1] allow the file owner to encrypt and sign the data at the client-side before storing it in the Cloud, [2] be able to audit operations performed in the Cloud, and [3] offer a scalable, user-friendly model for key management for efficient and secure data sharing.

3.1.1 Security Requirements:

- Data storage security
- Metadata storage security
- Owner controlled data sharing
- Data Integrity and Audit
- Masking ACL & Access Patterns
- Secure Search

3.3.2 Entities and Roles:



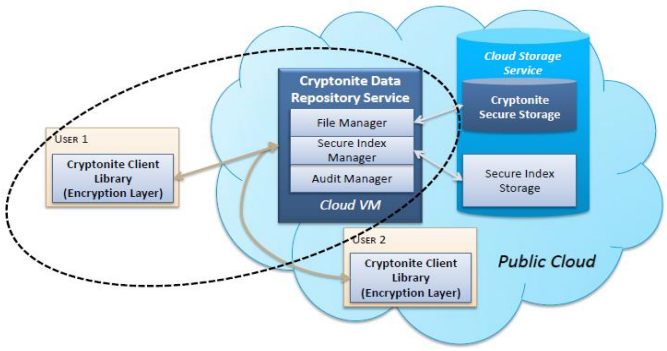
❖ User:
A community of users who require a secure storage repository for data storage and sharing

❖ Cloud Storage Service Provider :
- Provides the required persistent scalable storage space.
- Trusted with 'availability' (SLA)
- Not trusted with data security

❖ Secure Data Repository :
- Shared data repository

❖ ***Cryptonite :***
Not trusted with plain text data Partially trusted to perform requested operations (all operations should be verifiable).

4. Cryptonite Architecture:



4.1 Cryptographic Techniques – I

- Public Key Infrastructure(PKI)
- Digital Signatures
- Broadcast Encryption allows a user to encrypt their data such that it can be

$$K_{encr}^{shared} = f(K_{U_1}^{pub}, K_{U_2}^{pub}, \ldots)$$ subset of users.

$$F = encrypt(D, K_{encr}^{shared})$$

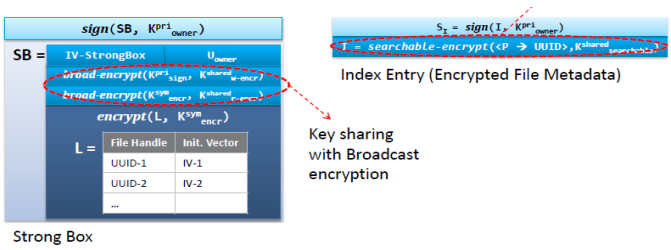
$$D = decrypt(F, K_{U_i}^{pri})$$

4.2 Cryptographic Techniques - II

- Lazy Revocation
  - A strategy of ***read*** access revocation, in which a file is not re-encrypted unless the file's contents change.
  - ***Key Rotation*** is used to enable forward Secrecy
- Searchable Encryption
  - Allows a user to search within an encrypted file given an appropriate "TrapDoor".
  - Without decrypting the entire file
  - Without revealing its contents to the searching entity
  - Current SE techniques lack ability to have fine grained access control over index entries and revocation strategies

4.2.1 Data Structures:

DATA File



5. Discussion

- Cryptonite protects data condentiality by performing client side encryption before data is stored in the repository
- "Trust but Verify": Signed acknowledgements let the end user prove unauthorised updates to his data.
- SSUID of Strongbox in plaintext

6. Related Work

- Commercial Tools
  - Microsoft Azure Storage, Amazon AWS/S3
    - Access Controlled by the providers
    - Providers have enough information to **decrypt the stored data.**
  - Nasuni Cloud Storage
    - Use cloud as storage backend.
    - More user control. But data **sharing granularity** is limited or requires **out-of-band key** exchange.
- Secure data storage in Distributed System
  - SiRiUS[14], PLUTUS[6] etc.
  - **Higher level of trust** on the Storage provider
- Data sharing through public Clouds
  - Cryptographic Cloud Storage[7], Cloud-Proof[8]
  - **Lack of file management** capabilities such as Secure Searchable Encryption

7. Future work

- Current BE techniques lack support for random access within an encrypted file.
- Write Serialization, Locking mechanism, Random file access to be addressed in future work.
- Next Step: Implementation and Deployment for USC microgrid Smart Grid initiative.

8. Conclusion

We have described the design and implementation of *Cryptonite*, a secure, performant data repository for Cloud platforms that offers sustainable key management features. Our empirical analysis shows that the optimizations that we propose do not have significant overhead and even perform better than the Azure .NET APIs for plaintext operations. It provides service compatibility with Azure's BLOB store and allows easy migration for Cloud data storage clients to incorporate security.

9. Acknowledgment

This material is based upon work supported by the references given below. The reviews and opinions of authors expressed herein are as per the literature survey of the topic mentioned above.

10. References

[1] Cloud Security Alliance, "Security Guidance for Critical Areas of Focus in Cloud Computing V2.1," Tech. Rep., 2009

[2] A. G. Kumbhare, Y. Simmhan, and V. K. Prasanna, "De-signing a secure storage repository for sharing scientific datasets using public clouds," in *Workshop on Data Intensive Computing in the Clouds*, 2011.

[3] Y. Simmhan, A. G. Kumbhare, B. Cao, and V. Prasanna, "An analysis of security and privacy issues in smart grid software architectures on clouds," in *IEEE CLOUD*, 2011

[4] S. Kamara and K. Lauter, "Cryptographic cloud storage," in Financial Cryptograpy and Data Security Conference, 2010.

[5] M. Backes, C. Cachin, and A. Oprea. Lazy revocation in cryptographic _le systems. Security in Storage Workshop, International IEEE, 0:1{11, 2005.

[6] G. Singh, S. Bharathi, A. Chervenak, E. Deelman, C. Kesselman, M. Manohar, S. Patil, and L. Pearlman. A metadata catalog service for data intensive applications. In Proceedings of the 2003 ACM/IEEE conference on Supercomputing, SC '03, pages 33-, New York, NY, USA, 2003. ACM.

[7] E.-J. Goh, H. Shacham, N. Modadugu, and D. Boneh. SiRiUS: Securing remote untrusted storage. In Network and Distributed System Security Conference (NDSS), 2003.

[8] M. Kallahalla, E. Riedel, R. Swaminathan, Q. Wang, and K. Fu. Plutus: Scalable secure _le sharing on untrusted storage. In USENIX Conference on File and Storage Technologies, 2003.