UPDATING POLICY DATA BY DELEGATION-AWARE ENCRYPTION STRATEGY

[1]SPVND SUNEETHA, [2]SPVND SUMALATHA, [3]B SARATH KUMARI

[1]Assistant Professor in CSE, [2]Assistant Professor in Mathematics, [3]Assistant Professor in Mathematics

[1]P.B.Siddhartha College Of Arts and Science, Vijayawada

[2,3]Andhra Loyola Institute of Engineering and Technology, Vijayawada-8

ABSTRACT:

A cryptographically enforced access controls for data facilitated in untrusted cloud is attractive for some users and associations. Be that as it may, planning proficient cryptographically enforced dynamic access control system in the cloud is as yet testing. In this work, we propose Crypt-DAC, a framework that gives reasonable cryptographic authorization of dynamic access control. A file is encoded by a symmetric key list which records a file key and a sequence of revocation keys. Accordingly, Crypt-DAC implements dynamic access control that gives proficiency, as it doesn't need expensive decryption/re encryption and uploading/re-uploading of large data at the administrator side, and security, as it promptly renounces access permissions. We use formalization structure and framework implementation to demonstrate the security and productivity of our development.

KEYWORDS: access control, cloud, revocation

1] INTRODUCTION:

In response to these security issues, numerous works [1], [4]–[9] have been proposed to support access control on untrusted cloud services by leveraging cryptographic primitives. Advanced cryptographic primitives are applied for enforcing many access control paradigms. For example, attribute based encryption (ABE) is a cryptographic counterpart of attribute-based access control (ABAC) model. However, previous works mainly consider static scenarios in which access control policies rarely change. The previous works incur high overhead when access control policies need to be changed in practice. At a first glance, the revocation of a user's permission can be done by revoking his access to the keys with which the files are encrypted. This solution, however, is not secure as the user can keep a local copy of the keys before the revocation. To prevent such a problem, files have to be re-encrypted with new keys. This requires the file owner to download the file, re-encrypt the file, and upload it back for the cloud to update the previous encrypted file, incurring prohibitive communication overhead at the file owner side

2] LITERATURE SURVEY:

2.1] Matteo Maffei *et al*

Cloud storage has rapidly become a cornerstone of many IT infrastructures, constituting a seamless solution for the backup, synchronization, and sharing of large amounts of data. Putting user data in the direct control of cloud service providers, however, raises security and privacy concerns related to the integrity of outsourced data, the accidental or intentional leakage of sensitive information, the profiling of user activities and so on. Furthermore, even if the cloud provider is trusted, users having access to outsourced files might be malicious and misbehave. These concerns are particularly serious in sensitive applications like personal health records and credit score systems. To tackle this problem, we present GORAM, a cryptographic system that protects the secrecy and integrity of outsourced data with respect to both an untrusted server and malicious clients, guarantees the anonymity and unlink ability of accesses to such data, and allows the data owner to share outsourced data with other clients, selectively granting them read and write permissions. GORAM is the first system to achieve such a wide range of security and privacy properties for outsourced storage. In the process of designing an efficient construction, we developed two new, generally applicable cryptographic schemes, namely, batched zero-knowledge proofs of shuffle and an accountability technique based on chameleon signatures, which we consider of independent interest. We implemented GORAM in Amazon Elastic Compute Cloud (EC2) and ran a performance evaluation demonstrating the scalability and efficiency of our construction.

2.2] Tao Jiang *et al*

The advent of the cloud computing makes storage outsourcing become a rising trend, which promotes the secure remote data auditing a hot topic that appeared in the research literature. Recently some research consider the problem of secure and efficient public data integrity auditing for shared dynamic data. However, these schemes are still not secure against the collusion of cloud storage server and revoked group users during user revocation in practical cloud storage system. In this paper, we figure out the collusion attack in the exiting scheme and provide an efficient public integrity auditing scheme

D.O.I. : 10.46528/DRSRJ.2021.V11I01.01

with secure group user revocation based on vector commitment and verifier-local revocation group signature. We design a concrete scheme based on the our scheme definition. Our scheme supports the public checking and efficient user revocation and also some nice properties, such as confidently, efficiency, countability and traceability of secure group user revocation

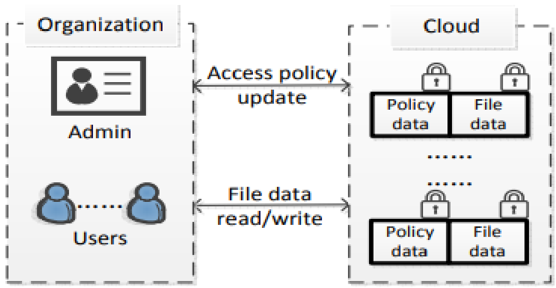
3] PROBLEM DEFINTION:

Gudes et al. [27] investigate cryptography to uphold hierarchy access control without considering dynamic policy scenarios. Akl et al. [28] propose a key task plan to simplify key management in hierarchical access control policy. Additionally, this work doesn't consider spolicy update issues. Afterward, Atallah et al. [29] propose a strategy that permits policy updates, however on account of revocation, all descendants of the influenced node in the access hierarchy of command should be updated, which includes high calculation and communication overhead.

4] PROPOSED APPROACH:

The proposed framework presents Crypt-DAC, a cryptographically upheld dynamic access control framework on un trusted cloud. Crypt DAC delegates the cloud to update encoded records in authorization revocations. In Crypt-DAC, a file is encrypted by a symmetric key list which records a file key and a sequence of revocation keys. In a revocation, the administrator uploads another revocation key to the cloud, which encrypts the file with another layer of encryption and updates the encoded key list accordingly.

5] SYSTEM ARCHITECTURE:



6] PROPOSED METHODOLOGY:

Cloud Server

D.O.I. : 10.46528/DRSRJ.2021.V11I01.01

The cloud service provider manages a cloud to provide data storage service. Data owners encrypt their data files and store them in the cloud for sharing with data consumers. To access the shared data files, data consumers download encrypted data files of their interest from the cloud and then decrypt them. The end user request will be processes based on the queue.

End User

The Cloud User who has a large amount of data to be stored in multiple clouds and have the permissions to access and manipulate stored data. The end user sends the request for corresponding file request and it will be processed in the cloud based on the queue and response to the end user.

Data Owner

The data owner uploads their data with its chunks in the cloud server. For the security purpose the data owner encrypts the data file's chunks and then store in the cloud. The Data owner can have capable of manipulating the encrypted data file. And the data owner can set the access privilege to the encrypted data file.

7] ALGORITHM:

Adjustable onion encryption and delayed de-onion encryption strategy:

Step 1: adjustable onion encryption strategy enable the administrator to define a tolerable bound for the file.

Step 2: Once the size of encryption layers reaches the bound, it can be made to not increase anymore by delegating encryption operations to the cloud.

Step 3: As a result, the administrator can flexibly adjust a tolerable bound for each file (according to file type, access pattern, etc.) to achieve a balance between efficiency and security.

Step4: delayed de-onion encryption strategy to periodically refresh the symmetric key list of the file and remove the bounded encryption layers over it through writing operations

Security mode

In this mode, a file fn is encrypted in a F tuple by a symmetric key list (k^0 , k^1 ,..., k^t) as follows:

D.O.I. : 10.46528/DRSRJ.2021.V11I01.01

$$\langle F,\ f_n,\ c\rangle$$

$$c = \mathsf{Enc}^{Sym}_{k^t}(\ldots\mathsf{Enc}^{Sym}_{k^1}(\mathsf{Enc}^{Sym}_{k^0}(f)))$$

When a user u accesses fn, u decrypts (k^0 , k^t , rpk_{fn} , t) from one of these FK tuples, recovers the revocation key sequence:

k^{i-1}←F-Dri(k^i , rpkfn) (2≤i≤t), and uses (k^0 , k^1 ,..., k^t) to decrypt the F tuple.

To complete the revocation, for each of the m files fn to which r has permissions, the administrator derives a new revocation key k $^{t+1}$ ← B-Dri(k^t , rsk_{fn}) and uploads k^{t+1} to the cloud provider. Upon receiving it, the cloud provider updates the F tuples of the m files as:

$$\langle F,\ f_n,\ \mathsf{Enc}^{Sym}_{k^{t+1}}(c)\rangle$$

Efficiency mode

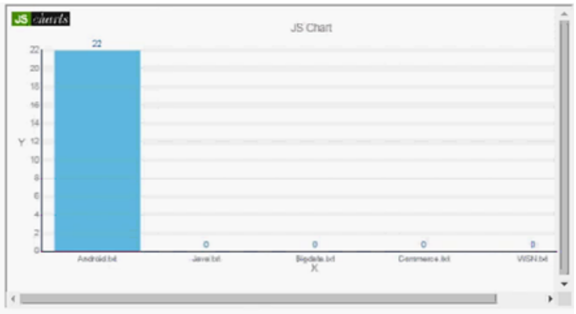
In this mode, a file fn in a F tuple is encrypted by a symmetric key list (k^0 , k^1 ,..., k^t) as follows:

$$\langle F,\ f_n,\ c\rangle$$

$$c = \mathsf{Enc}^{Sym}_{k^t}(\mathsf{Enc}^{Sym}_{k^0}(f))$$

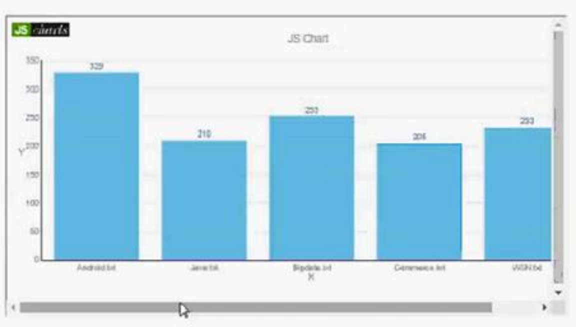
When a user u accesses fn, u decrypts (k^0 , k^t , rpk_{fn} , t) from one of these FK tuples and uses (k_0 , k^t) to decrypt the F tuple.

8] RESULTS:

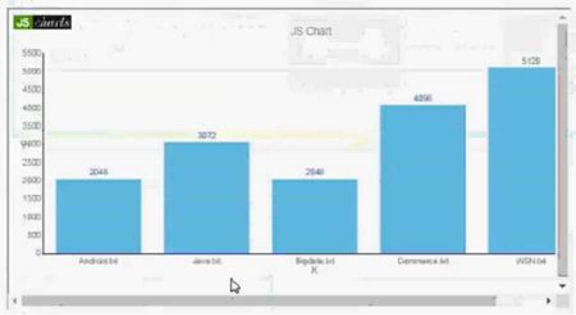


File rank details

D.O.I. : 10.46528/DRSRJ.2021.V11I01.01



Time **delay**



Throughput Details

9] CONCLUSION:

Crypt-DAC meets its goals using three techniques. In particular, we propose to delegate the cloud to update the policy data in a privacy-preserving manner using a delegation-aware encryption strategy. We propose to avoid the expensive re-encryptions of file data at the administrator side using a adjustable onion encryption strategy. In addition, we propose a delayed de-onion encryption strategy to avoid the file reading overhead. The theoretical analysis and the performance evaluation show that Crypt-DAC achieves orders of magnitude higher efficiency in access revocations while ensuring the same security properties under the honestbut-curious threat model compared with previous schemes.

10] REFERENCES:

[1] J. Bethencourt, A. Sahai, and B. Waters, Ciphertext-policy attribute based encryption, in IEEE S&P, 2007.

[2] X. Wang, Y. Qi, and Z. Wang, Design and Implementation of SecPod: A Framework for Virtualization-based Security Systems, IEEE Transactions on Dependable and Secure Computing, vol. 16, no. 1, 2019.

[3] J. Ren, Y. Qi, Y. Dai, X. Wang, and Y. Shi, AppSec: A Safe Execution Environment for Security Sensitive Applications, in ACM VEE, 2015.

[4] V. Goyal, A. Jain, O. Pandey, and A. Sahai, Bounded ciphertext policy attribute based encryption, in ICALP, 2008.

[5] V. Goyal, O. Pandey, A. Sahai, and B. Waters, Attribute-based encryption for fine-grained access control of encrypted data, in ACM CCS, 2006.

[6] J. Katz, A. Sahai, and B. Waters, Predicate encryption supporting disjunctions, polynomial equations, and inner products, in EUROCRYPT, 2008.

[7] S. Muller and S. Katzenbeisser, Hiding the policy in cryptographic access control, in STM, 2011.

[8] R. Ostrovsky, A. Sahai, and B. Waters, Attribute-based encryption with non-monotonic access structures, in ACM CCS, 2007.

[9] A. Sahai, and B. Waters, Fuzzy identity-based encryption, in EUROCRYPT, 2005.

[10] T. Ring, Cloud computing hit by celebgate, http://www.scmagazineuk. com/cloud-computing-hit-by-celebgate/article/370815/, 2015.

[11] X. Jin, R. Krishnan, and R. S. Sandhu, A unified attribute-based access control model covering DAC, MAC and RBAC, in DDBSec, 2012. [12] W. C. Garrison III, A. Shull, S. Myers, and, A. J. Lee, On the Practicality of Cryptographically Enforcing Dynamic Access Control Policies in the Cloud, in IEEE S&P, 2016.

[13] R. S. Sandhu, Rationale for the RBAC96 family of access control models, in ACM Workshop on RBAC, 1995.

[14] T. Jiang, X. Chen, Q. Wu, J. Ma, W. Susilo, and W. Lou, Secure and Efficient Cloud Data Deduplication With Randomized Tag, IEEE Trasactions on Data Forensics and Security, vol. 12, no. 3, 2017.

[15] M. Kallahalla, E. Riedel, R. Swaminathan, Q. Wang, K. Fu, Plutus: Scalable Secure File Sharing on Untrusted Storage, in USENIX FAST, 2003

D.O.I. : 10.46528/DRSRJ.2021.V11I01.01