ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

Enhanced Blind Signature based Secure Cloud Transactions

V.S.N Harshitha
M.Tech Student,
Department of CSE
Vignan'sNirula Institute of Technology and Science for Women, Pedapalakaluru, Guntur-522 005

P.Nagababu
Assistant Professor,
Department of CSE
Vignan'sNirula Institute of Technology and Science for Women, Pedapalakaluru, Guntur-522 005

ABSTRACT:

In distributed transactional database systems deployed over cloud servers, entities cooperate to form proofs of authorizations that are justified by collections of certified credentials. These proofs and credentials may be evaluated and collected over extended time periods under the risk of having the underlying authorization policies or the user credentials being in inconsistent states. It therefore becomes possible for policy-based authorization systems to make unsafe decisions that might threaten sensitive resources. In this paper, we highlight the criticality of the problem. We then define the notion of trusted transactions when dealing with proofs of authorization. Accordingly, we propose several increasingly stringent levels of policy consistency constraints, and present different enforcement approaches to guarantee the trustworthiness of transactions executing on cloud servers. We propose a Two-Phase Validation Commit protocol as a solution, which is a modified version of the basic Two-Phase Validation Commit protocols. We finally analyze the different approaches presented using both analytical evaluation of the overheads and simulations to guide the decision makers to which approach to use.

Index Terms—Cloud databases, authorization policies, consistency, distributed transactions, atomic commit protocol

INTRODUCTION:

Cloud computing has recently emerged as a computing paradigm in which storage and computation can be outsourced from organizations to next generation data centers hosted by companies such as Amazon, Google, Yahoo, and Microsoft. Such companies help free organizations from requiring expensive infrastructure and expertise in-house, and instead make use of the cloud providers to maintain, support, and broker access to high-end resources. From an economic perspective, cloud consumers can save huge IT capital investments and be charged on the basis of a pay-only-for-what-you-use pricing model. One of the most appealing aspects of cloud computing is its elasticity, which provides an illusion of infinite, on demand resources [1] making it an attractive environment for highly scalable, multitiered applications. However, this can create additional challenges for back-end, transactional database systems, which were designed without elasticity in mind. Despite the efforts of key-value stores like Amazon's SimpleDB, Dynamo, and Google's Bigtable to provide scalable access to huge amounts of data, transactional guarantees remain a bottleneck [2].To provide scalability and elasticity, cloud services often make heavy use of replication to ensure consistent performance and availability. As a result, many cloud services rely on the notion of eventual consistency when propagating data throughout the system. This consistency model is a variant of weak consistency that allows data to

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

be inconsistent among some replicas during the update process, but ensures that updates will eventually be propagated to all replicas. This makes it difficult to strictly maintain the ACID guarantees, as the "C" (consistency) part of ACID is sacrificed to provide reasonable availability [3].In systems that host sensitive resources, accesses are protected via authorization policies that describe the conditions under which users should be permitted access to resources. These policies describe relationships between the system principles, as well as the certified credentials that users must provide to attest to their attributes. In a transactional database system that is deployed in a highly distributed and elastic system such as the cloud, policies would typically be replicated—very much like data— among multiple sites, often following the same weak or eventual consistency model. It therefore becomes possible for a policy-based authorization system to make unsafe decisions using stale policies. Interesting consistency problems can arise as transactional database systems are deployed in cloud environments and use policy-based authorization systems to protect sensitive resources. In addition to handling consistency issues among database replicas, we must also handle two types of security inconsistency conditions. First, the system may suffer from policy inconsistencies during policy updates due to the relaxed consistency model underlying most cloud services. For example, it is possible for several versions of the policy to be observed at multiple sites within a single transaction, leading to inconsistent (and likely unsafe) access decisions during the transaction. Second, it is possible for external factors to cause user credential inconsistencies over the lifetime of a transaction [4].

For instance, a user's login credentials could be invalidated or revoked after collection by the authorization server, but before the completion of the transaction. In this paper, we address this confluence of data, policy, and credential inconsistency problems that can emerge as transactional database systems are deployed to the cloud. In doing so, we make the following contributions:

We formalize the concept of trusted transactions. Trusted transactions are those transactions that do not violate credential or policy inconsistencies over the lifetime of the transaction. We then present a more general term, safe transactions, that identifies transactions that are both trusted and conform to the ACID properties of distributed database systems We define several different levels of policy consistency constraints and corresponding enforcement approaches that guarantee the trustworthiness of transactions executing on cloud server. We propose a Two-Phase Validation Commit (2PVC) protocol that ensures that a transaction is safe by checking policy, credential, and data consistency during transaction execution. We carry out an experimental evaluation of our proposed approaches, and present a tradeoff discussion to guide decision makers as to which approach is most suitable in various situations.

Existing system

To protect user access patterns from a cloud data store, Williams et al. introduce a mechanism by which cloud storage users can issue encrypted reads, writes, and inserts. Further, Williams et al. propose a mechanism that enables un trusted service providers to support transaction serialization, backup, and recovery with full data confidentiality and correctness.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

- A dynamic consistency rationing mechanism that automatically adapts the level of consistency at runtime. Both of these works focus on data consistency, while our work focuses on attaining both data and policy consistency.
- Proofs of data possession have been proposed as a means for clients to ensure that service providers actually maintain copies of the data that they are contracted to host. In other works, data replications have been combined with proofs of retrieve ability to provide users with integrity and consistency guarantees when using cloud storage.
- CloudTPS is primarily concerned with providing consistency and isolation upon data without regard to considerations of authorization policies.
- This work proactively ensures that data stored at a particular site conforms to the policy stored at that site. If the policy is updated, the server will scan the data items and throw out any that would be denied based on the revised policy.
- The consistency of distributed proofs of authorization has previously been studied, though not in a dynamic cloud environment. This work highlights the inconsistency issues that can arise in the case where authorization policies are static, but the credentials used to satisfy these policies may be revoked or altered.
- The authors develop protocols that enable various consistency guarantees to be enforced during the proof construction process to minimize these types of security issues.

.

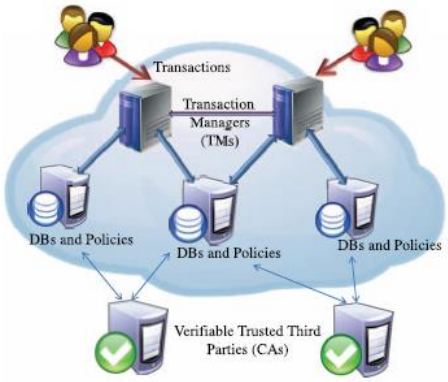

Fig. 1. Interaction among the system components.

Proposed System:

- In this paper highlight the criticality of the problem. It defines the notion of trusted transactions when dealing with proofs of authorization. Accordingly, it propose several increasingly stringent levels of policy consistency constraints, and present different enforcement approaches to guarantee the trustworthiness of transactions executing on cloud servers.
- It proposed a Two-Phase Validation Commit protocol as a solution, which is a modified version of the basic Two-Phase Validation Commit protocols.
- It finally analyze the different approaches presented using both analytical evaluation of the overheads and simulations to guide the decision makers to which approach to use.
- In this paper address this confluence of data, policy, and credential inconsistency problems that can emerge as transactional database systems are deployed to the cloud.
- This paper formalized the concept of trusted transactions. Trusted transactions are those transactions that do not violate credential or policy inconsistencies over the lifetime of the transaction.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

- It present a more general term, safe transactions, that identifies transactions that are both trusted and conforms to the ACID properties of distributed database systems.
- It defines several different levels of policy consistency constraints and corresponding enforcement approaches that guarantee the trustworthiness of transactions executing on cloud servers.

It proposed a Two-Phase Validation Commit (2PVC) protocol that ensures that a transaction is safe by checking policy, credential, and data consistency during transaction execution.

IMPLEMENTING SAFE TRANSACTIONS

Two-Phase Validation (2PV) Algorithm

A common characteristic of most of our proposed approaches to achieve trusted transactions is the need for policy consistency validation at the end of a transaction.That is, in order for a trusted transaction to commit, its TM has to enforce either view or global consistency among the servers participating in the transaction. Toward this, we propose a new algorithm called Two-Phase Validation.

Algorithm 1. Two-Phase Validation - 2PV(TM).

```
1  Send "Prepare-to-Validate" to all participants
2  Wait for all replies (a True/False, and a set of policy
   versions for each unique policy)
3  Identify the largest version for all unique policies
4  If all participants utilize the largest version for each
   unique policy
5     If any responded False
6        ABORT
7     Otherwise
8        CONTINUE
9  Otherwise, for all participants with old versions of
   policies
10    Send "Update" with the largest version number of
      each policy
11    Goto 2
```

Two-Phase Validate Commit Algorithm

The 2PV protocol enforces trusted transactions, but does not enforce safe transactions because it does not validate any integrity constraints. Since the Two-Phase Commit atomic protocol commonly used to enforce integrity constraints has similar structure as 2PV, we propose integrating these protocols into a Two-Phase Validation Commit protocol. 2PVC can be used to ensure the data and policy consistency requirements of safe transactions.

Algorithm 2. Two-Phase Validation Commit - 2PVC (TM).

```
1  Send "Prepare-to-Commit" to all participants
2  Wait for all replies (Yes/No, True/False, and a set of
   policy versions for each unique policy)
3  If any participant replied No for integrity check
4     ABORT
5  Identify the largest version for all unique policies
6  If all participants utilize the largest version for each
   unique policy
7     If any responded False
8        ABORT
9     Otherwise
10       COMMIT
11 Otherwise, for participants with old policies
12    Send "Update" with the largest version number of
      each policy
13    Wait for all replies
14    Goto 5
```

Using 2PV and 2PVC in Safe Transactions

Simulation Parameters

Parameter	Value(s)
Times of policies update	once during operations, once per participant join, or once at commit time
Disk read latency	1-3 ms
Disk write latency	12-20 ms
Authorization check delay	1-3 ms
Data integrity constraint verification	1-3 ms
Transaction size	Short: 8-15 operations, Medium: 16-30 operations, or Long: 31-50 operations

A variant of the basic 2PV protocol is used during the transaction execution. For view consistency, the TM needs to check the version number it receives from each server with that of the very first participating server. If they are different, the transaction aborts due to a consistency violation. At commit time, all the proofs will have been generated with consistent policies and only 2PC is invoked. In the global consistency case, the TM

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

needs to validate the policy versions used against the latest policy version known by the master policies server to decide whether to abort or not.

Environment and Setup

We used Java to implement each proof approach described in Section 3 with support for both view and global consistency. Although the approaches were implemented in their entirety, the underlying database and policy enforcement systems were simulated with parameters chosen according to Table 1. To understand the performance implications of the different approaches, we varied the

1. protocol used,
2. level of consistency desired,
3. frequency of master policy updates,
4. transaction length, and
5. number of servers available.

Our experimentation framework consists of three main components: a randomized transaction generator, a master policy server that controls the propagation of policy updates, and an array of transaction processing servers. Our experiments were run within a research lab consisting of 38 Apple Mac Mini computers.

RELATED WORK

Relaxed Consistency Models for the Cloud. Many database solutions have been written for use within the cloud environment. For instance, Amazon's Dynamo database[14]; Google's BigTable storage system [15]; Facebook's Cassandra [16]; and Yahoo!'s PNUTS [17]. The common thread between each of these custom data models is therelaxed notion of consistency provided to support massively parallel environments.

Such a relaxed consistency model adds a new dimension to the complexity of the design of large scale applications and introduces a new set of consistency problems [18]. The authors of [19] presented a model that allows queriers to express consistency and concurrency constraints on their queries that can be enforced by the DBMS at runtime. On the other hand, [20] introduces a dynamic consistency rationing mechanism that automatically adapts the level of consistency at runtime. Both of these works focus on data consistency, while our work focuses on attaining both data and policy consistency.

Reliable Outsourcing. Security is considered one of the major obstacles to a wider adoption of cloud computing. Particular attention has been given to client security as it relates to the proper handling of outsourced data. For example, proofs of data possession have been proposed as a means for clients to ensure that service providers actually maintain copies of the data that they are contracted to host [21]. In other works, data replication have been combined with proofs of retrievability to provide users with integrity and consistency guarantees when using cloud storage [22], [23].To protect user access patterns from a cloud data store, Williams et al. [24] introduce a mechanism by which cloud storage users can issue encrypted reads, writes, and inserts. Further, Williams et al. [25] propose a mechanism that enables untrusted service providers to support transaction serialization, backup, and recovery with full data confidentiality and correctness. This work is orthogonal to the problem that we focus on in this paper, namely consistency problems in policy-based database transactions.

Distributed Transactions. CloudTPS provides full ACID properties with a scalable transaction

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

manager designed for a NoSQL environment [26]. However, CloudTPS is primarily concerned with providing consistency and isolation upon data without regard to considerations of authorization policies.

Conclusions

Despite the popularity of cloud services and their wide adoption by enterprises and governments, cloud providers still lack services that guarantee both data and access control policy consistency across multiple data centers. In this paper, we identified several consistency problems that can arise during cloud-hosted transaction processing using weak consistency models, particularly if policy-based authorization systems are used to enforce access controls. To this end, we developed a variety of lightweight proof enforcement and consistency models—i.e., Deferred, Punctual, Incremental, and Continuous proofs, with view or global consistency—that can enforce increasingly strong protections with minimal runtime overheads. We used simulated workloads to experimentally evaluate implementations of our proposed consistency models relative to three core metrics: transaction processing performance, accuracy (i.e., global versus view consistency and recency of policies used), and precision (level of agreement among transaction participants). We found that high performance comes at a cost: Deferred and Punctual proofs had minimal overheads, but failed to detect certain types of consistency problems. On the other hand, high-accuracy models (i.e., Incremental and Continuous) required higher code complexity to implement correctly, and had only moderate performance when compared to the lower accuracy schemes. To better explore the differences between these approaches, we also carried out a tradeoff analysis of our schemes to illustrate how application-centric requirements influence the applicability of the eight protocol variants explored in this paper.

REFERENCES:

[1] M. Armbrust et al., "Above the Clouds: A Berkeley View of Cloud Computing," technical report, Univ. of California, Feb. 2009.

[2] S. Das, D. Agrawal, and A.E. Abbadi, "Elastras: An Elastic Transactional Data Store in the Cloud," Proc. Conf. Hot Topics in Cloud Computing (USENIX HotCloud '09), 2009.

[3] D.J. Abadi, "Data Management in the Cloud: Limitations and Opportunities," IEEE Data Eng. Bull., vol. 32, no. 1, pp. 3-12, Mar. 2009.

[4] A.J. Lee and M. Winslett, "Safety and Consistency in Policy-Based Authorization Systems," Proc. 13th ACM Conf. Computer and Comm. Security (CCS '06), 2006.

[5] M. Myers, R. Ankney, A. Malpani, S. Galperin, and C. Adams, "X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - Ocsp," RFC 2560, http://tools.ietf.org/html/rfc5280, June 1999.

[6] E. Rissanen, "Extensible Access Control Markup Language (Xacml) Version 3.0," http://docs.oasis-open.org/xacml/3.0/ xacml-3.0-core-spec-os-en.html, Jan. 2013.

[7] D. Cooper et al., "Internet x.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile," RFC 5280, http://tools.ietf.org/html/rfc5280, May 2008.

[8] J. Li, N. Li, and W.H. Winsborough, "Automated Trust Negotiation Using Cryptographic Credentials," Proc. 12th ACM Conf. Computer and Comm. Security (CCS '05), Nov. 2005.

[9] L. Bauer et al., "Distributed Proving in Access-Control Systems