IJMTARC – VOLUME – IV – ISSUE - 16 - DEC 2016 ISSN: 2320-1363

TO SECURE THE CLOUD DATA AND CHECK THE PERFORMACE OF THE CLOUD DATA USING ACCURACY AND PRECISION

Mrs.P.SIndhuja, Mr.G.RANGITH

Abstract—In distributed transactional database systems deployed over cloud servers, entities cooperate to form proofs of authorization 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 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.

1 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-whatyou-use pricing model. One of the most

appealing aspects of cloud computing is its elasticity, which provides an illusion of infinite, ondemand resources [1] making it an attractive environment for highly-scalable, multi-tiered 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 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 be inconsistent among some replicas during the update process, but ensures that updates will eventually be propagated to all replicas. Thismakes it difficult to strictly maintain the ACID guarantees, as the 'C' (consistency) part of ACID is sacrificed to provide reasonable availability . 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 principals, 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 amongst 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 (Sec. 2). • We define several different levels of policy consistency constraints and corresponding enforcement approaches that guarantee the trustworthiness of transactions being executed on cloud servers (Sec. 3). • 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 (Sec. 4). • We carry out an experimental evaluation of our proposed approaches (Sec. 5), and present a trade-off discussion to guide decision makers as to which approach is most suitable in various situations (Sec 6). Finally, Sec. 7 describes previous related work, while Sec. 8 presents our conclusions.

2 SYSTEM ASSUMPTION AND PROBLEMDEFINATION

2.1 System Model

Fig. 1 illustrates the interaction among the components in our system. We assume a cloud infrastructure consisting of a set of S servers, where each server is responsible for hosting a subset D of all data itemsDbelonging to a specific application domain (D ⊂D). Users interact with the system by submitting queries or update requests encapsulated in ACID transactions. A transaction is submitted to a Transaction Manager (TM) that coordinates its execution. Multiple TMs could be invoked as the system workload increases for load balancing, but each transaction is handled by only one TM. We denote each transaction as T = q1,q2,...,qn, where qi ∈ Q is a single query/update belonging to the set of all

queries Q. The start time of each transaction is denoted by α(T), and the time at which the transaction finishes execution and is ready to commit is denoted by ω(T). We assume that queries belonging to a transaction execute sequentially, and that a transaction does not fork sub-transactions. These assumptions simplify our presentation, but do not affect the correctness or the validity of our consistency definitions. Let P denote the set of all authorization policies, and let Psi(D) denote the policy that server si uses to protect data item D. We represent a policy P as a mapping P : S×2D → 2R×A×N that associates a server and a set of data items with

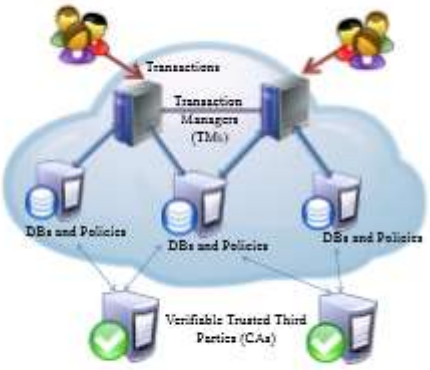

Fig. 1. Interaction among the system components

a set of inference rules from the set R, a policy administrator from the set A, and a version number. We denote by C the set of all credentials, which are issued by the Certificate Authorities (CAs) within the system. We assume that each CA offers an online method that allows any server to check the current status of credentials that it has issued [5]. Given a credential ck ∈C, α(ck) and ω(ck) denote issue and expiration times of ck, respectively. Given a function m : Q → 2D that identifies the data items accessed by a particular query, a proof of authorization for query qi evaluated at server sj at time tk is a tuple hqi, sj,Psj(m(qi)),tk,Ci, where C is the set of credentials presented by the querier to satisfy Psj(m(qi)). In this paper, we use the function eval : F ×TS →B to denote whether a proof f ∈F is valid at time t ∈ TS. To

enhance the general applicability of the consistency models developed in this paper, the above formalism is intentionally opaque with respect to the policy and credential formats used to implement the system. 2.2 Problem Definition

Since transactions are executed over time, the state information of the credentials and the policies enforced by different servers are subject to changes at any instance of time, therefore it becomes important to introduce precise definitions for the different consistency levels that could be achieved within a transactions lifetime. These consistency models strengthen the trusted transaction definition by defining the environment in which policy versions are consistent relative to the rest of the system. Before we do that, we define a transaction's view in terms of the different proofs of authorization evaluated during the lifetime of a particular transaction.

Definition 1: (View) A transaction's view VT is the setof proofs of authorization observed during the lifetime of a transaction [α(T),ω(T)] and defined as VT = {fsi | fsi = hqi, si,Psi(m(qi)),ti,Ci∧qi ∈ T}. _ Following from Def. 1, a transaction's view is built incrementally as more proofs of authorization are being evaluated by servers during the transaction execution. We now present two increasingly more powerful definitions of consistencies within transactions.

Definition 2: (View Consistency) A view VT = {hqi, si,Psi(m(qi)),ti,Ci,...,hqn, sn,Psn(m(qn)),tn,Ci} is view consistent, or φ-consistent, if VT satisfies a predicate φ-consistent that places constraints on the versioning of the policies such that φ-consistent(VT) ↔∀i,j : ver(Psi) = ver(Psj) for all policies belonging to the same administrator A, where function ver is defined as ver : P →N. _ With a view consistency model, the policy versions should be internally consistent across all servers executing the transaction. The view consistency model is weak in that the policy version agreed upon by the subset of servers within the transaction may not be the latest policy version v. It may be the case that a server outside of the S servers has a policy that belongs to the same administrative domain and with a version v0 > v. A more strict consistency model is the global consistency and is defined as follows.

Definition 3: (Global Consistency) A view VT = {hqi, si,Psi(m(qi)),ti,Ci,...,hqn,

sn,Psn(m(qn)),tn,Ci} is global consistent, or ψ-consistent, if VT satisfies a predicate ψ-consistent that places constraints on the versioning of the policies such that ψ-consistent(VT) ↔ ∀i : ver(Psi) = ver(P) for all policies belonging to the same administrator A, and function ver follows the same aforementioned definition, while ver(P) refers to the latest policy version. _ With a global consistency model, policies used to evaluate the proofs of authorization during a transaction execution among S servers should match the latest policy version among the entire policy set P, for all policies enforced by the same administrator A. Given the above definitions, we now have a precise vocabulary for defining the conditions necessary for a transaction to be asserted as "trusted".

Definition 4: (Trusted Transaction) Given a transaction T ={q1,q2,...,qn}and its corresponding view VT, T is trusted iff ∀fsi∈VT : eval(fsi,t), at some time instance t : α(T) ≤ t ≤ ω(T) ∧ (φ-consistent(VT) ∨ ψ-consistent(VT)) _ Finally, we say that a transaction is safe if it is a trusted transaction that also satisfies all data integrity constraints imposed by the database management system. A safe transaction is allowed to commit, while an unsafe transaction is forced to rollback.

3TRUSTEDTRANSACTION ENFORCEMENT

In this section, we present several increasingly stringent approaches for enforcing trusted transactions. We show that each approach offers different guarantees during the course of a transaction. Fig. 2 is a graphical depiction of how these approaches could be applied to a transaction running across three server, and will be referenced throughout this section. In this figure, dots represent the arrival time of a query to some server, stars indicate the times at which a server validates a proof of authorization, and dashed lines represent view- or globally-consistency policy synchronization among servers.

3.1 Deferred Proofs of Authorization

Definition 5: (Deferred Proofs of Authorization) Given a transaction T and its corresponding view VT, T is trusted under the Deferred proofs of authorization approach, iff at commit time ω(T), ∀fsi∈VT : eval(fsi,ω(T)) ∧ (φ-consistent(VT) ∨ ψconsistent(VT)) _Deferred proofs present an optimistic approach with relatively weak

authorization guarantees. The proofs of authorization are evaluated simultaneously only at commit time (using either view or global consistency from Defs. 2 and 3) to decide whether the transaction is trusted.

3.2 Punctual Proofs of Authorization

Definition 6: (Punctual Proofs of Authorization) Given a transaction T and its corresponding view VT, T is trusted under the Punctual proofs of authorization approach, iff at any time instance ti : $\alpha(T) \leq ti \leq \omega(T)$ $\forall fsi \in VT$: $eval(fsi,ti) \wedge eval(fsi,\omega(T)) \wedge (\varphi\text{-consistent}(VT) \vee \psi\text{-consistent}(VT))$ _ Punctual proofs present a more proactive approach in which the proofs of authorization are evaluated instantaneously whenever a query is being handled by a server. This facilitates early detections of unsafe transactions which can save the system from going into expensive undo operations. All the proofs of authorization are then re-evaluated at commit time to ensure that policies were not updated during the transaction in a way that would invalidate a previous proof, and that credentials were not invalidated. Punctual proofs do not impose any restrictions on the freshness of the policies used by the servers during the transaction execution. Consequently, servers might falsely deny or allow access to data. Thus, we propose two more restrictive approaches that enforce some degree of consistency among the participating servers each time a proof is evaluated.

3.3 Incremental Punctual Proofs of Authorization

Before we define the Incremental Punctual proofs of authorization approach, we define a view instance, which is a view snapshot at a specific instance of time.

Definition 7: (View Instance) A view instance VT ti $\subseteq$ VT is defined as VT ti = $\{fsi \mid fsi = hqi, si, PA\ si(m(qi)), t, Ci \in VT \wedge t \leq ti\}$, $\forall t,ti : \alpha(T) \leq t \leq ti \leq \omega(T)$. _ Informally, a view instance VT ti is the subset of all proofs of authorization evaluated by servers involved in transaction T up until the time instance ti.

Definition 8: (Incremental Punctual Proofs of Authorization) Given a transaction T and its corresponding view VT, T is trusted under the Incremental Punctual proofs of authorization approach, iff at any time instance ti : $\alpha(T) \leq ti \leq \omega(T)$, $\forall fsi \in VT\ ti$: $eval(fsi,ti) \wedge (\varphi\text{-consistent}(VT\ ti) \vee \psi\text{-consistent}(VT\ ti))$ _ Incremental Punctual

proofs develop a stronger notion of trusted transactions, as a transaction is not allowed to proceed unless each server achieves the desired level of the policy consistency with all previous servers. This implies that all participating servers will be forced to have a consistent view with the first executing server unless a newer policy version shows up at a later server, in which case the transaction aborts. For view consistency, no consistency check at commit time is required, since all participating servers will be view consistent by commit time. On the other hand, the global consistency condition necessitates another consistency check at commit time to confirm that the policies used have not become stale during the window of execution between the last proofs of authorization and commit time.

3.4 Continuous Proofs of Authorization

We now present the least permissive approach which we call Continuous proofs of authorization.

Definition 9: (Continuous Proofs of Authorization) A transaction T is trusted under the Continuous approach, iff $\forall 1 \leq i \leq n \forall 1 \leq j \leq i$: eval(fsi,ti)$\wedge$eval(fsj,ti) $\wedge$ (φ-consistent(VT ti) $\vee$ ψ-consistent(VT ti)) at any time instance t : $\alpha(T) \leq ti \leq \omega(T)$ _ In Continuous proofs, whenever a proof is evaluated, all previous proofs have to be re-evaluated if a newer version of the policy is found at any of the participating servers. At commit time, Continuous proofs behave similarly to Incremental Punctual proofs. In contrast with the Incremental Punctual proofs, if later executing servers are using newer policy versions, all previous servers must (i) update their policies to be consistent with the newest one, and (ii) re-evaluate their proofs of authorization using the newer policies. In the case of global consistency, all servers will be forced to use the latest policy version at all times. Therefore, we consider this variant of our approaches to be the most strict approach of all giving the best privacy and consistency guarantees. The decision of which approach to adopt is likely to be a strategic choice made independently by each application. As with any trade-off, the stronger the security and accuracy given by an approach, the more the system has to pay in terms of implementation and messages exchange overheads.

6 Conclusion

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 article, 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 light-weight 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 vs. 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 trade-off analysis of our schemes to illustrate how application-centric requirements influence the applicability of the eight protocol variants explored in this article.

7 References

[1] M. Armbrust et al., "Above the clouds: A berkeley view of cloud computing," University of California, Berkeley, Tech. Rep., Feb. 2009. [

2] S. Das, D. Agrawal, and A. El Abbadi, "Elastras: an elastic transactional data store in the cloud," in USENIX HotCloud, 2009.

[3] D. J. Abadi, "Data management in the cloud: Limitations and opportunities," IEEE Data Engineering Bulletin, Mar. 2009.

[4] A. J. Lee and M. Winslett, "Safety and consistency in policy-based authorization systems," in ACM CCS, 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, Jun. 1999, http://tools.ietf.org/html/rfc5280.

[6] E. Rissanen, "extensible access control markup language (xacml) version 3.0," Jan. 2013, http://docs.oasis-open.org/xacml/3.0/xacml-3.0core-spec-os-en.html.

[7] D. Cooper et al., "Internet x.509 public key infrastructure certificate and certificate revocation list (crl) profile," RFC 5280, May 2008, http://tools.ietf.org/html/rfc5280. [8] J. Li, N. Li, and W. H. Winsborough, "Automated trust negotiation using cryptographic credentials," in ACM CCS, Nov. 2005. [9] L. Bauer et al., "Distributed proving in access-control systems," in Proc. of the IEEE Symposium on Security and Privacy, May 2005.

[10] J. Li and N. Li, "OACerts: Oblivious attribute based certificates," IEEE TDSC, Oct. 2006.

[11] J. Camenisch and A. Lysyanskaya, "An efficient system for nontransferable anonymous credentials with optional anonymity revocation," in EUROCRYPT, 2001. [12] P. K. Chrysanthis, G. Samaras, and Y. J. Al-Houmaily, "Recovery and performance of atomic commit processing in distributed database systems," in Recovery Mechanisms in Database Systems. PHPTR, 1998.

[13] M. K. Iskander, D. W. Wilkinson, A. J. Lee, and P. K.Chrysanthis, "Enforcing policy and data consistency of cloud transactions," in IEEE ICDCS-SPCC, 2011

Inform., Comput., Commun. Security, 2006, pp. 297– 302.

[20] S. S. M. Chow, S.-M. Yiu, and L. C. K. Hui, "Efficient identity based ring signature," in Proc. 3rd Int. Conf. Appl. Cryptography Netw. Security, 2005, vol. 3531, pp. 499–512.

[21] R. Cramer, I. Damgard, and B. Schoenmakers, "Proofs of partial knowledge and simplified design of witness hiding protocols," in Proc. 14th Annu. Int. Cryptol. Conf. Adv. Cryptol., 1994, vol. 839, pp. 174–187.

[22] R. Cramer and V. Shoup, "Signature schemes based on the strong RSA assumption," in Proc. ACM Conf. Comput. Commun. Security, 1999, pp. 46–51.

[23] Y. Dodis, A. Kiayias, A. Nicolosi, and V. Shoup, "Anonymous identification in Ad Hoc groups," in Proc. Int. Conf. Theory

Appl. Cryptographic Techn., 2004, vol. 3027, pp. 609–626.

[24] A. L. Ferrara, M. Green, S. Hohenberger, and M

Author's profile:

Mr.G.Ranjith received M.Tech degree from JNTUH, Hyderabad. He is currently working as Assistant professor, Department of CSE, in Nalgonda Institute of Technology &Sceince, Nalgonda, Telangana, India. His interests includes Web Technologies, Java Programming, Data Base Management Systems.

Mrs.P.Sinduja B.Tech Degree from Nalgonda Institute of Technology & Science in Nalgonda. She is currently pursuing M.tech Degree in Computer Science and Engineering specialization in Nalgonda Instituite of Technology & Science in Nalgonda, Telangana, India.