ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

An ISO 3297: 2007 Certified Organization *Vol.3, Special Issue 4, April 2015*

National Conference On Emerging Trends in Information, Digital & Embedded Systems (NC'e-TIDES -15)

Organized by

Dept. of ECE, Annamacharya Institute Of Technology & Sciences, Rajampet, Andhra Pradesh-516126, India held on 28th February 2015

Providing Secure Transactions in Cloud Databases

Chamanchi Naresh

M.Tech, Department of CSE, A.I.T.S., Rajampet, Kadapa(dist), Andhra Pradesh, India

ABSTRACT: Cloud Servers are maintaining distributed transactional databases. In these transactions all entities are cooperate to form Proofs of Authorizations which are justified by collections of certified credentials. These authorization proofs and certified credentials may be evaluated and collected over extended time periods under the risk of having underlying authorizations polices and user credentials being in inconsistent states. It becomes possible to make unsafe decisions that might threaten sensitive resource for policy based authorization systems. Mainly cloud computing provides many benefits in terms of low cost and accessibility of data. We cannot directly implement any security mechanisms in cloud platform due to varying of cloud service and deployment models. The major factor is providing a security for database transactions in cloud environment; users often store sensitive information in cloud databases that may be untrusted. In this paper we highlighted the criticality of problem that is the notion of untrusted transactions when dealing with the proof of authorizations in cloud environment. According to that we propose several levels of consistency constraints that are guarantee the safe and trustworthiness of transactions which are executing on cloud servers. Here we propose a Two Phase Validation Commit Protocol as a solution for analysing the different approaches presented using both analytical evaluation and simulation to the decision makers which one to use.

KEYWORDS: Cloud databases, authorization policies, consistency, distributed transactions, atomic commit protocol.

I. INTRODUCTION

Now a days cloud computing is an important computing paradigm in which storage and computation can be performed. Only limited companies are providing Cloud Computing such as Amazon, Google, IBM, Microsoft, and Yahoo etc. Such companies help free organizations from requiring expensive infrastructure. Cloud providers are 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 model.

Cloud computing is having mainly three servicemodels and four deployment models. Those are Platform as a Service (Paas), Software as a Service (SaaS) and Infrastructure as a Service (IaaS), the deployment models are Public Cloud, Private Cloud, Hybrid Cloud, and Community Cloud.

One of the most important aspects of Cloud Computing is its elasticity and scalability, which provides infinite services to cloud consumers, on demand resources making it an efficient and attractive environment for highly scalable, multi-tiered applications. 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 likeAmazon'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 all updates will eventually be propagated to all replicas. This will makes 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 principles, as well as the certified credentials that users must provide to attest to their attributes. In a traditional

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

An ISO 3297: 2007 Certified Organization Vol.3, Special Issue 4, April 2015

National Conference On Emerging Trends in Information, Digital & Embedded Systems (NC'e-TIDES -15)

Organized by

Dept. of ECE, Annamacharya Institute Of Technology & Sciences, Rajampet, Andhra Pradesh-516126, India held on 28th February 2015

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 dataamong multiple sites, often following the same weak or eventual consistency model. It therefore becomes possible for a policy-based authorization system to make unsafedecisions using stale policies.

Consistency problems can arise intransactional database systems in cloud environments and use policy-based authorization systems toprotect sensitive resources. In addition to handling consistency issues among database replicas, we must 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. 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 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 servers.
- 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 are most suitable in various situations.

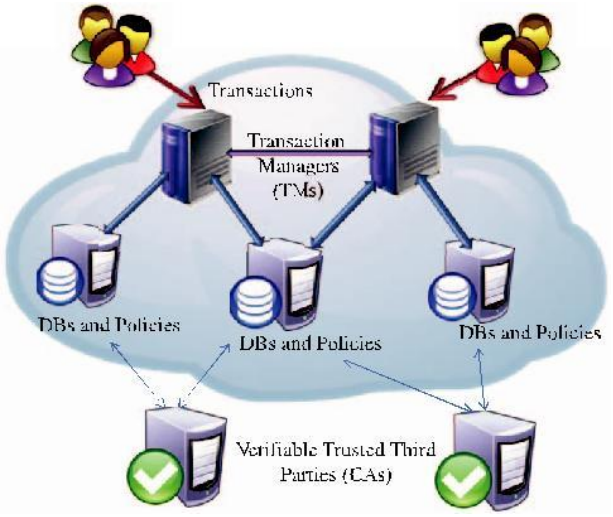


Fig. 1 Interaction among system components.

The above Fig. 1 illustrates the interaction betweendifferent components in a cloud. In a cloud infrastructure consisting of a set of S servers, where each server is responsible for hosting a subset *D* of all data items D belonging to a specific application domain ($D \subset \mathsf{D}$). 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.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

An ISO 3297: 2007 Certified Organization Vol.3, Special Issue 4, April 2015

National Conference On Emerging Trends in Information, Digital & Embedded Systems (NC'e-TIDES -15)

Organized by

Dept. of ECE, Annamacharya Institute Of Technology & Sciences, Rajampet, Andhra Pradesh-516126, India held on 28th February 2015

II. RELATED WORK

Relaxed Consistency Models for the Cloud: Many database solutions were written for use within the cloud environment. For instance, Amazon's Dynamo database, Google's BigTable storage system; Facebook's Cassandra; and Yahoo!'s PNUTS. The common thread between each of these custom data models is the relaxed 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. The authors of 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, 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 as a one of the major important aspect 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. In other works, data replications have been combined with proofs of irretrievability to provide users with integrity and consistency guaranteeswhen using cloud storage.

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

There has also been recent work that focuses on providing some level of guarantee to the relationship between data and 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. It is obvious that this will lead to an eventually consistent state where data and policy conform, but this work only concerns itself with local consistency of a single node, not with transactions that span multiple nodes.

Distributed Authorization: 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. These consistency guarantees are similar to our notions of safe transactions. However, our work also addresses the case in which policies—in addition to credentials—may be altered or modified during a transaction.

III. PROPOSED ALGORITHM

A. *Two-Phase Validation (2PV) Algorithm:*

A common characteristic of our proposed approach to achieve trusted transactions 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.

As the name implies, 2PV operates in two phases: collection and validation. During collection, the TM first sends a Prepare-to-Validate message to each participant server. In response to this message, each participant 1)

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

An ISO 3297: 2007 Certified Organization Vol.3, Special Issue 4, April 2015

National Conference On Emerging Trends in Information, Digital & Embedded Systems (NC'e-TIDES -15)

Organized by

Dept. of ECE, Annamacharya Institute Of Technology & Sciences, Rajampet, Andhra Pradesh-516126, India held on 28th February 2015

evaluates the proofs for each query of the transaction using the latest policies it has available and 2) sends a reply back to the TM containing the truth value (TRUE/FALSE) of those proofs along with the version number and policy identifier for each policy used. Further, each participant keeps track of its reply (i.e., the state of each query) whichincludes the id of the TM (TMid), the id of the transaction (Tid) to which the query belongs, and a set of policy versions used in the query's authorization (vi, pi).

Once the TM receives the replies from all the participants, it moves on to the validation phase. If all polices are consistent, then the protocol honours the truth value whereany FALSE causes an ABORT decision and all TRUE cause a CONTINUE decision. In the case of inconsistent policies, the TM identifies the latest policy and sends an Update message to each out-of-date participant with a policy identifier and returns to the collection phase. In this case, the participants 1) update their policies, 2) re-evaluate the proofs and, 3) send a new reply to the TM. Algorithm 1 shows the process for the TM.

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

B. *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. 2PVCcan be used to ensure the data and policy consistency requirements of safe transactions. Specifically, 2PVC will evaluate the policies and authorizations within the first, voting phase. That is, when the TM sends out a Prepare-to-Commit message for a transaction, the participant server has three values to report 1) the YES or NO reply for the satisfaction of integrity constraints as in 2PC, 2) the TRUE or FALSE reply for the satisfaction of the proofs of authorizations as in 2PV, and 3) the version number of the policies used to build the proofs (vi, pi) as in 2PV.

The process given in Algorithm 2 is for the TM under view consistency. It is similar to that of 2PV with the exception of handling the YES or NO reply for integrity constraint validation and having a decision of COMMIT rather than CONTINUE. The TM enforces the same behaviour as 2PV in identifying policies inconsistencies and sending the Update messages. The same changes to 2PV can be made here to provide global consistency by consulting the master policies server for the latest policy version (Step 5).

The process given in Algorithm 2 is for the TM under view consistency. It is similar to that of 2PV with the exception of handling the YES or NO reply for integrity constraint validation and having a decision of COMMIT rather than CONTINUE. The TM enforces the same behaviour as 2PV in identifying policies inconsistencies and sending the Update messages. The same changes to 2PV can be made here to provide global consistency by consulting the master policies server for the latest policy version (Step 5).

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

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

An ISO 3297: 2007 Certified Organization Vol.3, Special Issue 4, April 2015

National Conference On Emerging Trends in Information, Digital & Embedded Systems (NC'e-TIDES -15)

Organized by

Dept. of ECE, Annamacharya Institute Of Technology & Sciences, Rajampet, Andhra Pradesh-516126, India held on 28th February 2015

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

IV. PSEUDO CODE

Step 1: Begin a Transaction
Step 2: TM sends a "Prepare toCommit "message to all particular participate Server.
Step 3: Get reply message from particular participate server.
    If NO
        Abort
    Else
        Continue
Step 4: If any inconsistent policies
    send out-of-date message to each participant server
    else
        Continue
Step 5: Execute Transaction
Step 6: End Transaction

V. SIMULATION RESULTS

We measure the average execution time of the *shortest*successfully committed transactions (denoted t_s), which occurswhen there are no policy changes, and the average execution time of the *longest* successfully committed transactions(denoted t_f), which occurs when policy changes force reevaluations of the proofs of authorization or multiple roundsof 2PV are invoked (e.g., in Continuous proofs). Essentially, t_f captures the cost of recovering from a conflict. In the case of Continuous proofs, the worst case is when a policy change is detected each time a new server joins the execution of a transaction. The average transaction execution time to terminate (abort or commit) for Deferred, Punctual, and Continuous proofs can be computed using the following equation, where $P_u$represents the probability of a policy update:

$$t=t_s(1-p_u)+t_fP_u \quad (1)$$

As opposed to the other proofs of authorization, in Incremental Punctual proofs, if a policy change is detected during the execution of a transaction, the transaction will abort regardless if it is using view or global consistency. Thereforeto compute the average execution time, we assume that each aborted transaction is re-executed once to successful commit, with all servers using consistent policies. This assumption approximates the cost for rolling back the aborted transactions. We use the following equation to compute the average transaction execution time:

$$t=(t_f + t_s)P_u+t_s(1-P_u) \quad (2)$$

Wheret_f denotes the measured average time of the quickest aborted transactions among the simulation runs, and t_s denote the average time of the successfully committed transactions.

Using Equations 1 and 2, we plot Figures 3 and 4 to show our simulation results for both the LAN and the WAN arrangements respectively. Each figure shows the execution time of the committed transaction (y-axis) as the probability of the policy update changes (x-axis). The figures contrast between the four different approaches for proofs of authorization each with the two validation modes, namely, view and global consistency. The figures show different

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

An ISO 3297: 2007 Certified Organization Vol.3, Special Issue 4, April 2015

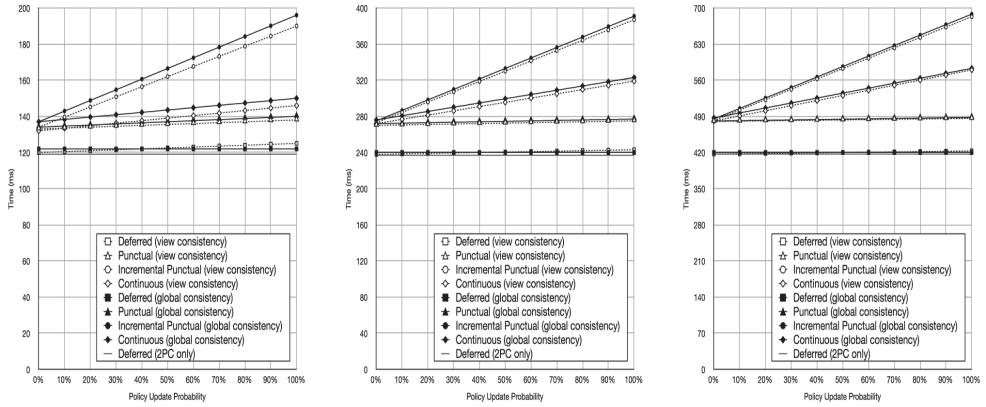
National Conference On Emerging Trends in Information, Digital & Embedded Systems (NC'e-TIDES -15)

Organized by

Dept. of ECE, Annamacharya Institute Of Technology & Sciences, Rajampet, Andhra Pradesh-516126, India held on 28th February 2015

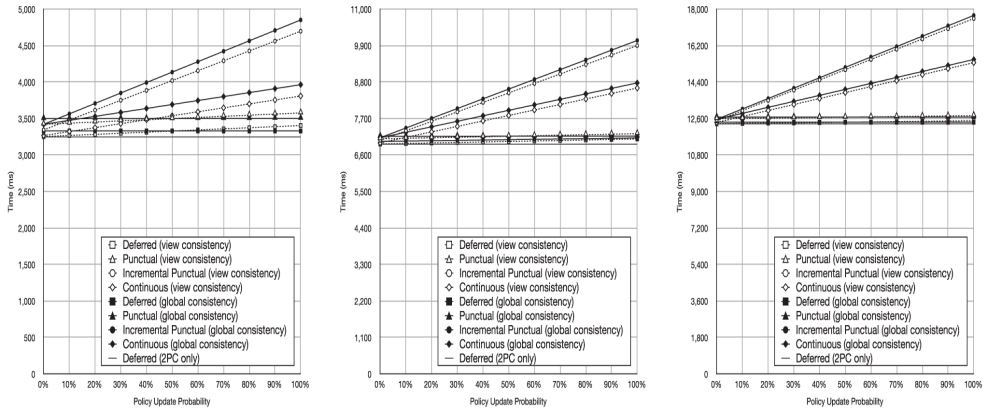
transactions length: (a) short transactions involve 8–15 operations running on up to 5 servers, (b) medium transactions involve 16–30 operations running on up to 15 servers, and (c) long transactions involve 31–50 operations running on up to 25 servers. For each case, and as a baseline, we measured the transaction execution time when transactions execute without any proof of authorization and are terminated using the basic 2PC (shown in figures as a solid line referring to deferred 2PC only). In all cases, the average transaction execution time of deferred proofs with 2PVC was effectively the same as the baseline indicating that 2PVC has negligible overhead over the basic 2PC.

The relative performance of the different proofs of authorization is consistent throughout the different experiments. From the figures, we can conclude that the deferred proofs have the best performance of all, as the transaction operations are allowed to proceed without interruption until commit time. Of course, proofs of authorization failing at commit time will force the transaction to go into a potentially expensive rollback. That will not be the case with the other schemes, as the proofs are evaluated earlier during the execution of the transactions and the rollback process of aborted transactions involves fewer operations.



(a) Short Transactions (8–15 operations) (b) Medium Transactions (16–30 operations) (c) Long Transactions (31–50 operations)

Fig 2. Results for LAN Experiments



(a) Short Transactions (8–15 operations) (b) Medium Transactions (16–30 operations) (c) Long Transactions (31–50 operations)

Fig 3. Results for WAN Experiments.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

An ISO 3297: 2007 Certified Organization Vol.3, Special Issue 4, April 2015

National Conference On Emerging Trends in Information, Digital & Embedded Systems (NC'e-TIDES -15)

Organized by

Dept. of ECE, Annamacharya Institute Of Technology & Sciences, Rajampet, Andhra Pradesh-516126, India held on 28th February 2015

Punctual proofs come next in terms of performance. The minor difference between Punctual and Deferred proofs is because Punctual proofs incur the cost for the local authorization checks each of which is in the range of 3-5 ms. Both Deferred and Punctual proofs are on average insensitive to the probability of policy updates (as realized from the graph slope). This is due to the fact that both schemes only enforce consistency at commit time.

Incremental Punctual proofs show the worst performance of all schemes and are the most sensitive to the probability of policy updates. This is due to the fact that Incremental Punctual proofs using either view or global consistency have the risk of aborting and re-executing each time a policy update is encountered. As the policy update probability increases, the performance of Incremental Punctual is severely penalized.

Continuous proofs show better performance than the Incremental Punctual approach, but are worse than the Deferred and Punctual approaches. Just as with Incremental Punctual, the performance of Continuous proofs suffers as the probability of policy update increases, as with each policy update all previously evaluated proofs will go through a re-evaluation phase using the 2PV protocol. A final observation is that in most cases, global consistency proofs are slightly slower than view consistency. This extra latency comes from the additional communication round between TM and the master policy server to retrieve the latest policy version. Global consistency proofs were faster than view consistency ones in the few cases when the latest policy happens to match policy used by all participating servers andas a result all servers skip the re-evaluation step of 2PVC.

VI. CONCLUSION AND FUTURE WORK

The popularity of cloud services and their wide adoption by enterprises, governments and different organizations, cloud providers still lack services that guarantee both data and access control policy consistency across multiple data centres. 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. At the end, we developed variety of consistency modelsi.e., Deferred, Punctual, Incremental, and Continuous proofs, with view or global consistencythat can increasingly strong protections with minimum runtime overheads.

We used simulated workloads to experiment the evaluate implementations of our proposed consistency models relative to three main core metrics such as transaction processing performance, accuracy, and precision for providing secure transactions in cloud databases. We found high performance comes at a cost of deferred and punctual proofs had minimum overheads. Finally 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-offanalysis of our schemes to illustrate how application-centric requirements influence the applicability of the eight protocol variants explored in this article. Finally we develop a good performance, accuracy and precision with implementing of Scalability and Elasticity for database transactions in cloud.

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," Proc. IEEE Symp. Security and Privacy May 2005.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

An ISO 3297: 2007 Certified Organization *Vol.3, Special Issue 4, April 2015*

National Conference On Emerging Trends in Information, Digital & Embedded Systems (NC'e-TIDES -15)

Organized by

Dept. of ECE, Annamacharya Institute Of Technology & Sciences, Rajampet, Andhra Pradesh-516126, India held on 28th February 2015

10. J. Li and N. Li, "OACerts: Oblivious Attribute Based Certificates," IEEE Trans. Dependable and Secure Computing, vol. 3, no. 4, pp. 340-352, Oct.-Dec. 2006.
11. J. Camenisch and A. Lysyanskaya, "An Efficient System for Non-Transferable Anonymous Credentials with Optional Anonymity Revocation," Proc. Int'l Conf. Theory and Application of Cryptographic Techniques: Advances in Cryptology (EUROCRYPT '01), 2001.
12. P.K. Chrysanthis, G. Samaras, and Y.J. Al-Houmaily, "Recovery and Performance of Atomic Commit Processing in Distributed Database Systems," Recovery Mechanisms in Database Systems, Prentice Hall PTR, 1998.
13. M.K. Iskander, D.W. Wilkinson, A.J. Lee, and P.K. Chrysanthis, "Enforcing Policy and Data Consistency of Cloud Transactions," Proc. IEEE Second Int'l Workshop Security and Privacy in Cloud Computing (ICDCS-SPCCICDCS-SPCC), 2011.
14. G. DeCandia et al., "Dynamo: Amazons Highly Available Key-Value Store," Proc. 21st ACM SIGOPS Symp. Operating Systems Principles (SOSP '07), 2007.
15. F. Chang et al., "Bigtable: A Distributed Storage System for Structured Data," Proc. Seventh USENIX Symp. Operating System Design and Implementation (OSDI '06), 2006.

BIOGRAPHY

Chamanchi Naresh,M.Tech in the Computer Science & Engineering, College of Annamacharya Institute of Technology and Sciences, Rajampet. He received Bachelor of Technology (B.Tech) degree in 2009 from N.B.K.R.I.S.T., Vidyanagar, Vakadu, Nellore District, A.P., and India. His research interests is Cloud Computing.