A Service Oriented Transaction Invoicing Approach in Cloud Computing

Poral Nagaraja,
Dept of CS&E, SJMIT,
Chitradurga-577501.

MeenaSajjan G P,
PG student,
Dept of CS&E, SJMIT,
Chitradurga-577501.

ABSTRACT
Cloud resource usage tracking and invoicing in a trusted manner are inevitable and critical for the cloud service provider. The credibility of the service is measured in terms of accuracy in invoicing for the service consumed. In the existing system the limitations are, complexity, computational overhead and no way to validate the usage. Here, we propose OSIRIS: The consumption based efficient invoicing of the service oriented transaction in cloud computing. This system addresses all the existing concerns. It uses a concept called cloud notary authority and is responsible for accuracy in invoicing. This acts as an interface between cloud service provider and user and usage can be verified on either side. We have trusted SLA monitoring mechanism too that is built on trusted platform module called I-Mon.The performance evaluation confirms that the overall latency of OSRIS invoicing transactions is much shorter than the latency of the existing leading methodology. OSIRIS guarantees identical security features as a PKI [10].

General Terms
Invoicing, trusted monitoring, transaction integrity,cloud resources

Keywords
OSRIS, SLA Compliance, usage tracking,µ-contract

1. INTRODUCTION
The legacy model of IT Resources/Services delivery is hosting it over locally, whereas Cloud Computing does it over the internet, e.g. Amazon EC2, S3 [1], and Microsoft Azure [2]. The services delivered using Cloud Computing comprise of applications, services and the infrastructure required to deliver those resources/services. These services are purchased by the Cloud Consumer on a need basis and thus can avoid capital investment on procuring hardware or software to deliver the services/resources. This is a flexible way of doing things since the services can be catered on demand/future-demand basis without any major change in IT landscape. The extent of hardware/software virtualization is the core of Cloud Computing model and is the key driver for reducing IT costs with transparency. The billing for the subscribed Cloud Services involves many complications such as monitor SLA [3] and ensure credible, easy, cost effective, and mutually verifiable billing system. The current system is composed of PKI based complex, high computational overhead thus resulting high latency in billing response of the system [14].

Currently, the cloud consumer invoice is generated based on the resources consumed by pay-as-you-go pricing model for the SLA made between CSP and Cloud consumer/customer. Invoicing does not indicate whether the CSP conforms to SLA when the services are consumed. Also a systematic forgery vulnerable logging does not exist, that can be used for verification of usage invoicing on both the side - Customer and CSP [4], [5].
A secure and non-obstructive billing system called OSIRIS is proposed which uses the concept of a Cloud Notary Authority for the supervision of billing. OSRIS addresses the concerns of existing system and has provisions for a) reduced computational overhead, b)SLA monitoring is provided in a trusted manner and c) Accurate, consistent and mutually verifiable invoicing.

2. SYSTEM ARCHITECTURE
The billing transaction is initiated by a service check-in for starting a cloud service session and terminated by a service check-out for finalizing the service session. A µ-contract message is transmitted with each billing transaction. A µ-contract is a data structure that contains a hashed value of a billing context and the hash chain element of each entity. The Cloud Notary Authority (CNA) is the only entity can decrypt both the µ-contract from the CSP and the µ-contract of the user, the CNA serves as a third party to verify the consistency of the billing context between the user and the CSP. Fig.1 shows the overall process of the billing transactionwith our billing system. The main steps are as follows:

- The user generates and sends a cloud resource request message to CSP.
- The CSP generates a digital signature called µ-contract using an element from its hash chain.
- The user generates a digital signature called µ-contract using an element from its hash chain.
- The user sends the combined µ-contract of its own and of the CSP to the CNA.
- The CNA verifies the µ-contract from the user, and generates mutually verifiable binding information of the user and the CSP to ensure the consistency of the µ-contract.
- The billing process is completed when the user and the CSP receive confirmation from the CNA.
- I-Mon of the user's cloud resource transmits authentication data of the I-Mon to the CNA.
- For service check-out, I-Mon sends a report of the SLA monitoring results to the CNA.

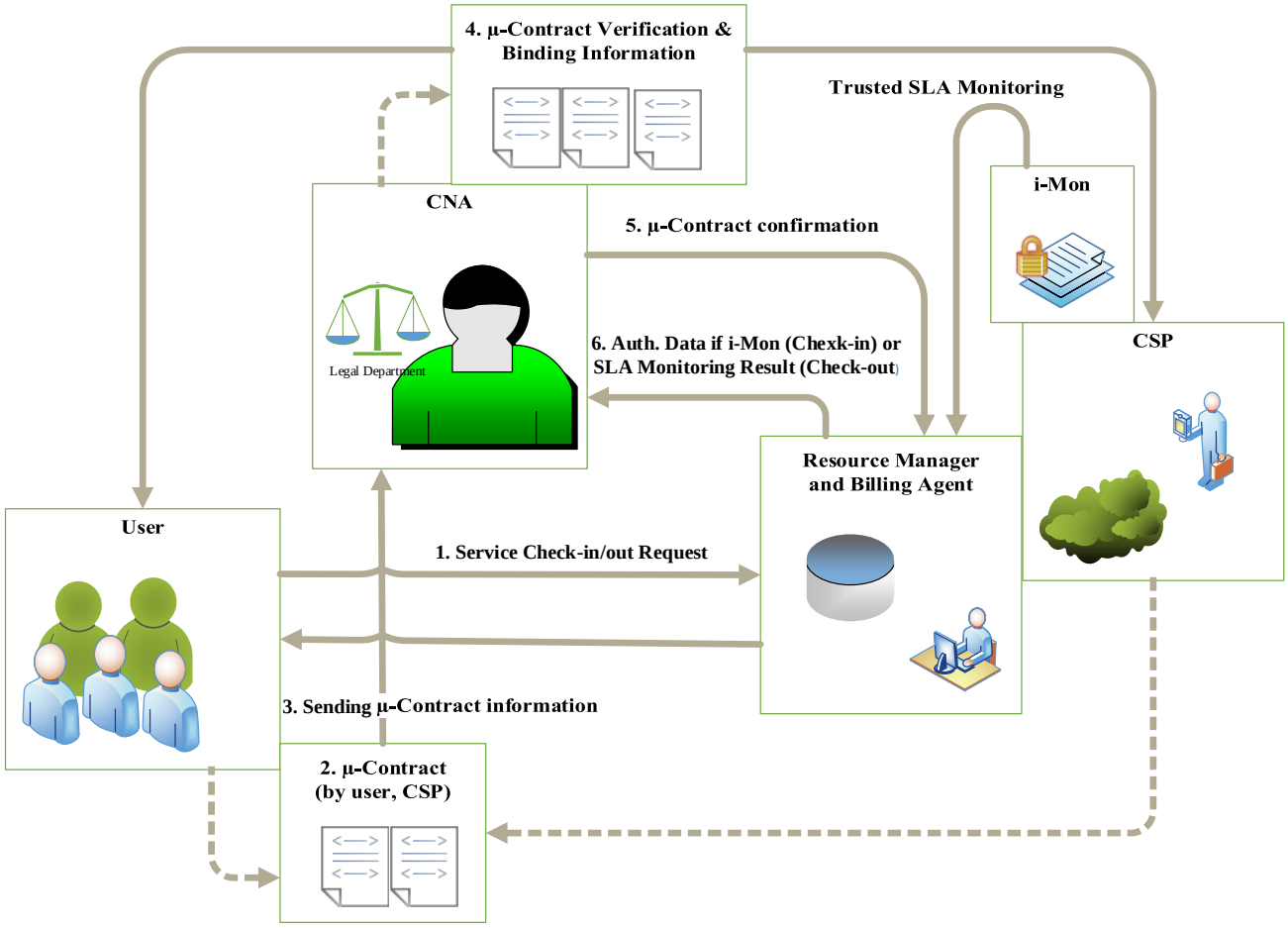

Figure 1.System Architecture - OSIRIS

3. RELATED WORKS

Extensive studies[6], [7], [8], [9],[11], [12]on existing system reveal many aspects on security vulnerabilities and limitations in the system, the latency in invoicing, the verification of each transactions against the SLA compliance, handling disputes etc. We dictates the pros and cons of different billing systems in terms of their security level and billing overhead based on the vast studies and experimental results part of our due-diligence work on assessing of existing system with the future needs and current limitations.

3.1 Existing System

The billing system with limited security concerns and the micro-payment based billing system require a relatively low level of computational complexity. This is clear visible from the studies of the micro-payment based schemes such as,MiniPay [20], PayWord [21], e-coupons [22]and NetPay [23]. The average billing latency for billing system with limited security is 4.06 ms for micro-payment based billing system, it is 4.07ms. Nevertheless, these systems are inadequate in terms of transaction integrity, non-repudiation and trusted SLA monitoring. In spite of the consensus, a PKI-based billing system offering a high level of security through two security functions, unlike trustworthySLAmonitoring, the security comes at the price of extremely complex[11], [12], [13],[15].

Table 1. Summary of relevant works

System	Transaction Integrity	Non-Repudiation	Trusted SLA Monitoring	Billing Latency
Billing System with limited security	No	No	No	Avg. 4.06 ms
Micro-payment based billing system	Yes	No	No	Avg. 4.70 ms
PKI- based billing system	Yes	Yes	No	Avg. 82.51 ms
OSIRIS	Yes	Yes	Yes	Avg. 4.89 ms

PKI enhanced billing frameworks have been studied using the market models DGAS [11], SGAS [12], and GridBank [13]. The study shows that PKI operations with the average billing latency of 82.51ms.Consequently, when a PKI-based billing system is used in cloud computing environment, the high computational complexity causes high deployment cost and high operational overload as the PKI operations are performed by the user and the CSP.

There are several studies have been conducted for SLA monitoring - resource monitoring [17], data flow monitoring [18], prediction of SLA violations [19]. These methodologies are used in the design I-Mon for OSIRIS, and are nowhere present in the existing system [16].

3.2 Proposed System

In this paper, we propose a secure and non-obstructive billing system called OSIRIS as a remedy for the above mentioned limitations. The system uses a novel concept of a cloud notary authority (CNA) for the supervision of billing. The CNA generates mutually verifiable binding information that can be used to resolve future disputes between a user and CSP in a computational efficient way. Further, scalability and fault tolerance is done in banking side by providing security for bill payment which is a web service leading to faster time to market, minimal computational cost, accurate, consistent and competitive pricing. The average billing latency of OSIRIS is 4.89ms.

4. PROPOSED BILLING PROTOCOL

This section describes the end to end transactions of the proposed system.

4.1 Notations used in OSIRIS

Table 2. Notations used in OSIRIS

Definition of the entity symbols	
c	Cloud Service Provider (CSP)
u	User
n	Cloud Notary Authority (CNA)
m	SLA Monitoring Module (S-Mon)
Definition of the Message symbols	
$K_{\alpha,\beta}$	Shared Key between α and β
PK_{α}	Public Key of α
SKα	Private Key of α
H(M)	Hash result for message M
→ Usage sequence	
Hash Chain: C0 - C1 - C2 - ... - Cn, H(Cn) =Cn-1	
← Chain generation sequence	
$C_{\alpha,n}$	nth element of the hash chain of α
T_s	Time-Stamp
N_{α}	Nonce value for preventing replay attack by α
S	Stipulation context of billing transaction
{M}K	M encrypted by K
{M}PK	M encrypted by public Key
{M}SK	Digital signature for M by private key

4.2 Transactions in OSIRIS

The end to end transactions in OSIRIS has 3 states as shown in Fig 2.

State 1: Mutual Authentication When the user First time accesses the CSP, PKI-based authentications are performed by the user, the CSP, and the CAN and they share the following keys for all authentications:

- CSP ↔ CNA: Kc,n
- User ↔ CNA: Ku,n
- User ↔ CSP: Ku,c

State 2 (Hash Chain Generation)
A hash chain of length 'N' will be generated and seeded (for seed value Cu;N , Cc;N , and Cn;N) by N times by each CSP, CNA, and user to obtain the final hash keys re (Cu;0, Cc;0, and Cn;0). The user and CSP commit the final hash (Cu;0 and Cc;0) by digitally signing and send it to CNA for registration. CNA then generates its final hash (Cn;0). A μ-contract is created for billing transactions once the hash chain is successfully committed.

State 3 (Billing Transaction)

- State 3 .1: (Billing Transaction) Service Check in.
  User sends a check-in request to CSP for the intended cloud service (Message 3-1).

CSP sends back S-Stipulation (a service invoice and an SLA information) and u-contract to the user (Message 3-2).
User generates a notary request by combining the µ-contract-CSP and the µ-contract-User message and sends to CNA (Message 3-3)
CNA verifies the u-contracts present in Message 3-3, if they are identical CNA send the confirmation message to user and CSP (Message 3-4).

- State 3.2: (Billing Transaction) Service Check out.
  When the user sends Check-out message, I-Mon sends SLA monitoring results to CNA (Message 3-5). CNA does the SLA compliance check and impose penalties upon violation of SLA

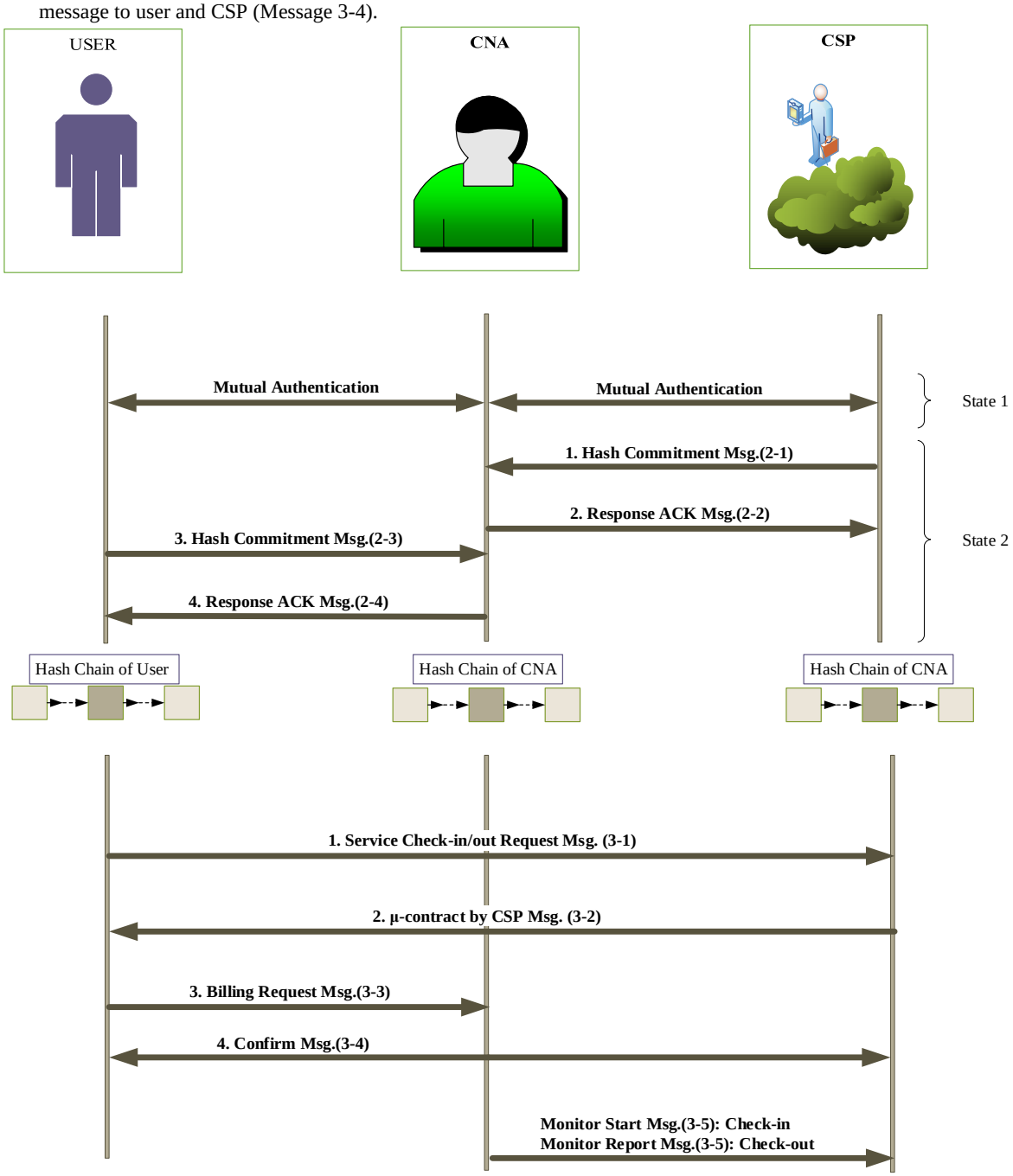

Figure 2. Transactions in OSIRIS

4.3 I-Mon: SLA-Monitor

I-Mon is deployed into computing resources of CSP to provide a forgery-resistive SLA measuring and logging mechanism in a black-box (BB) manner. Thus, even the administrator of the CSP cannot modify or falsify the logged data.

I-Mon is tightly coupled with the transactions described in section 4.2. The following are the steps involved in SLA monitoring.

1. I-Mon is initialized and verified during the service check-in transaction (via State 3-1).

2. During the service session, I-Mon monitors the SLA compliance with regard to the user's cloud resources.

3. I-Mon generates and sends the SLA monitoring result to the CNA on the event of check-out transaction (via State 3-2).

4. CNA records the service interval and monitoring result from I-Mon.

The TPM [24] and the TXT [25] are the two hardware-based mechanisms used in I-Mon. The TPM is designed for the purpose of secure storage and remotely determining the trustworthiness of a platform. TXT is a secure execution mechanism by Intel called a measured launch environment (MLE) it enables a verified execution code in a secure memory region. I-Mon uses the following fundamental technologies:

Platform integrity measurement: Trusted execution of I-Mon is guaranteed by TPM. TPM has a set of built-in 160-bit platform configuration registers (PCRs). The MLE uses the PCR's characteristics. MLE can compare the PCR value with a reference value to ensure that only a verified execution code is invoked in a secure memory region. To enable the CNA to verify the platform status, the TPM provides a Quote() function, which uses a TPM private key called an attestation identity (AIK) to return a digital signature of the current PCR values. The AIK is created inside the TPM and protected by the TPM so that Quote() provides proof that the output of Quote() was generated on the platform.

Secure storage with the TPM: The TPM encrypts the input data with a TPM key and specified PCR values. I-Mon can instruct TPM to decrypt and encrypt PCR values.

Execution integrity with the TPM: The TPM has built-in support for a monotonic counter and has a mechanism that creates a signature of the current tick value of the TPM. The tick data include a signature of the current tick value and its update cycle. These functions are utilized in our verification mechanism. The verification mechanism enables the CNA to determine whether the I-Mon has been executed without a block or a data loss; it also determines when SLA violations occur with the tick value.

4.4 Billing Verification

I-Mon provides a forgery-resistive SLA measuring and logging mechanism in a black-box (BB) manner. Thus, even the administrator of the CSP cannot modify or falsify the logged data. CNA stores all binding information and BB corresponding to a billing transaction at its local repository. This is an XML-data structure and called NBL.

The verification module has three hash modules: The User-Verifier, the CNA-Verifier, and the CSP-Verifier. The CNA-Verifier verifies the integrity of the stipulation (S) from the user or the CSP by comparing the stipulation with the binding information of the CNA. In addition, the CNA-Verifier can check the correctness of the BB by comparing the H(S)of the NBL with the H(S) of the BB. The User-Verifier and the CSP-Verifier check the correctness of a billing transaction asserted by the user and the CSP, respectively.

For example, if a CSP asserts that a user repudiates a certain billing transactions, the CSP can submit a claim for justice to the CNA, drawing attention to the stipulation (S) included in the corresponding μ-contract-CSP. The CNA then uses the CNA-verifier to verify the claim. If the claim is correct, the CNA then demands to see the stipulation (S) used to generate the μ-contract-User. The CNA uses the User-Verifier and the CSP-Verifier to derive the hash value. Any discrepancy between the output of the hash function and the stored data of the NBL proves that either the user or the CSP has modified the stipulation of the relevant billing.

5. CONCLUSION

The aim was to provide a scalable, secure, efficient, low computational overhead, mutually verifiable invoicing system for service oriented transactions measured against SLA between the CSP and end user. This is a go solution as it addresses the concerns and vulnerabilities in the existing system and hence cloud consumers and service providers are likely to accept this methodology.

6. ACKNOWLEDGEMENT

The authors would like to thank principal and staffs S.J.M.I.T.Chitradurga for their support. This work was supported by S.J.M.Vidyapeetha ®.

7. REFERENCES

[1] Amazon Web Services, "Amazon Elastic Compute Cloud EC2,Simple Storage Service," http://aws.amazon.com/ec2, http://aws.amazon.com/s32, Apr. 2011.

[2] Microsoft, "Microsoft, Windows Azure Platform," http://www.microsoft.com/windowsazure, 2010.

[3] M. Armbrust and A.E. Fox, "Above the Clouds: A Berkeley Viewof Cloud Computing," Technical Report UCB/EECS-2009-28,Electrical Engineering and Computer Sciences Dept., Univ. ofCalifornia, Berkeley, Feb. 2009.

[4] N. Santos, K.P. Gummadi, and R. Rodrigues, "Towards TrustedCloud Computing," Proc. Conf. Hot Topics in Cloud Computing(HotCloud), 2009.

[5] R.T. Snodgrass, S.S. Yao, and C. Collberg, "Tamper Detection inAudit Logs," Proc. 30th Int'l Conf. Very Large Data Bases (VLDB'04), pp. 504-515, 2004.

[6] L. Cornwall, M. Craig, R. Byrom, and R. Cordenonsib, "APEL:An Implementation of Grid Accounting Using R-GMA," Proc. UKE-Science All Hands Conf., Sept. 2005.

[7] F. Tannenbaum, L. Foster, and Tuecke, "Condor-G: A ComputationManagement Agent for Multi-Institutional Grids," ClusterComputing, vol. 5, pp. 237-246, 2002.

[8] O.-K. Kwon, J. Hahm, S. Kim, and J. Lee, "GRASP: A GridResource Allocation System Based on OGSA,"

Proc. IEEE 13th Int'lSymp. High Performance Distributed Computing, pp. 278-279, 2004.

[9] "Tivoli: Usage and Accounting Manager," IBM press release, 2009.

[10] PKIX Working Group, http://www.ietf.org/html.charters/pkixcharter.html, 2008.

[11] A. Guarise, R. Piro, and A. Werbrouck, "Datagrid AccountingSystem—Architecture—v1.0," technical report, EU DataGrid, 2003.

[12] P. Gardfill, E. Elmroth, L. Johson, O. Mulmo, and T. Sandholm,"Scalable Grid-Wide Capacity Allocation with the SweGridAccounting System (SGAS)," Concurrency Computation: PracticeExperience, vol. 20, pp. 2089-2122, Dec. 2008.

[13] A. Barmouta and R. Buyya, "Gridbank: A Grid AccountingServices Architecture (GASA) for Distributed Systems Sharingand Integration," Proc. 17th Int'l Symp. Parallel and DistributedProcessing (IPDPS '03), pp. 22-26, 2003.

[14] G. von Voigt and W. Muller, "Comparison of Grid AccountingConcepts for D-Grid," Proc. Cracow Grid Workshop, pp. 459-466,Oct. 2006.

[15] NexR, "iCube Cloud Computing and Elastic-Storage Services,"http://www.nexr.co.kr/, Mar. 2011.

[16] H. Rajan and M. Hosamani, "Tisa: Toward Trustworthy Servicesin a Service-Oriented Architecture," IEEE Trans. Services Computing,vol. 1, no. 4, pp. 201-213, Oct.-Dec. 2008.

[17] S. Meng, L. Liu, and T. Wang, "State Monitoring in CloudDatacenters," IEEE Trans. Knowledge and Data Eng., vol. 23, no. 9,pp. 1328-1344, Sept. 2011.

[18] C. Olston and B. Reed, "Inspector Gadget: A Framework forCustom Monitoring and Debugging of Distributed Dataflows,"Proc. ACM SIGMOD Int'l Conf. Management of Data (SIGMOD '11),pp. 1221-1224, 2011.

[19] P. Leitner, A. Michlmayr, F. Rosenberg, and S. Dustdar, "Monitoring, Prediction and Prevention of SLA Violations in Composite Services," Proc. IEEE Int'l Conf. Web Services (ICWS),pp. 369-376, 2010.

[20] A. Herzberg and H. Yochai, "MiniPay: Charging per Click on theWeb," Proc. Selected Papers from the Sixth Int'l Conf. World WideWeb, pp. 939-951, 1997.

[21] R. Rivest, A. Shamir, "PayWord and MicroMint: two simple micropayment schemes", 1996 International Workshop on Security Protocols, Lecture Notes in Computer Science, vol. 1189, Springer, pp. 69–87

[22] V. Patil, R.K. Shyamasundar, "An efficient, secure and delegable micro-payment system", 2004 IEEE International

[23] X. Dai and J. Grundy, "NetPay: An Off-Line, DecentralizedMicro-Payment System for Thin-Client Applications," ElectronicCommerce Research Applications, vol. 6, pp. 91-101, Jan. 2007.

[24] S. Pearson and B. Balacheff, Trusted Computing Platforms: TCPATechnology in Context. Prentice Hall Professional, 2003.

[25] "Intel Trusted Execution Technology, Hardware-Based Technologyfor Enhancing Server Platform Security," white paper, Intel,2010.