THEMIS: A Mutually Verifiable Billing System

for the Cloud Computing Environment

**Introduction:**

Cloud computing is an important transition that makes change in service oriented computing technology. Cloud service provider follows pay-as-you-go pricing approach which means consumer uses as many resources as he need and billed by the provider based on the resource consumed. CSP give a quality of service in the form of a service level agreement. For transparent billing, each billing transaction should be protected against forgery and false modifications. Although CSPs provide service billing records, they cannot provide trustworthiness. It is due to user or CSP can modify the billing records. In this case even a third party cannot confirm that the user’s record is correct or CSPs record is correct. To overcome these limitations we introduced a secure billing system called THEMIS. For secure billing system THEMIS introduces a concept of cloud notary authority (CNA). CNA generates mutually verifiable binding information that can be used to resolve future disputes between user and CSP. This project will produce the secure billing through monitoring the service level agreement (SLA) by using the SMon module. CNA can get a service logs from SMon and stored it in a local repository for further reference. Even administrator of a cloud system cannot modify or falsify the data.

**Scope of the Project:**

Scope of the project is to provide a high securable and non obstructive billing system. Central Nodal Authority (CNA) generates the bill with binding information. The process, which involves a generation of mutually verifiable binding information among all the involved entities on the basis of a one-way hash chain, is computationally efficient for a thin client and the CSP. So even administrator of a cloud system cannot modify or falsify the data.

**Literature Survey:**

**Title:** Above the clouds: A Berkeley view of cloud computing

**Author:** M. Armbrust and A. E. Fox

**Year:** 2009

**Description:**

Cloud Computing, the long-held dream of computing as a utility, has the potential to transform a large part of the IT industry, making software even more attractive as a service and shaping the way IT hardware is designed and purchased. Developers with innovative ideas for new Internet services no longer require the large capital outlays in hardware to deploy their service or the human expense to operate it. They need not be concerned about over provisioning for a service whose popularity does not meet their predictions, thus wasting costly resources, or under provisioning for one that becomes wildly popular, thus missing potential customers and revenue. Moreover, companies with large batch-oriented tasks can get results as quickly as their programs can scale, since using 1000 servers for one hour costs no more than using one server for 1000 hours. This elasticity of resources, without paying a premium for large scale, is unprecedented in the history of IT. Cloud Computing refers to both the applications delivered as services over the Internet and the hardware and systems software in the datacenters that provide those services. The services themselves have long been referred to as Software as a Service (SaaS). The datacenter hardware and software is what we will call a Cloud. When a Cloud is made available in a pay-as-you-go manner to the general public, we call it a Public Cloud; the service being sold is Utility Computing. We use the term Private Cloud to refer to internal datacenters of a business or other organization, not made available to the general public. Thus, Cloud Computing is the sum of SaaS and Utility Computing, but does not include Private Clouds. People can be users or providers of SaaS, or users or providers of Utility Computing. We focus on SaaS Providers (Cloud Users) and Cloud Providers, which have received less attention than SaaS Users.

**Title:** Towards trusted cloud computing

**Author:** N. Santos, K. P. Gummadi, and R. Rodrigues

**Year:** 2009

**Description:**

Cloud computing infrastructures enable companies to cut costs by outsourcing computations on-demand. However, clients of cloud computing services currently have no means of verifying the confidentiality and integrity of their data and computation. To address this problem we propose the design of a trusted cloud computing platform (TCCP). TCCP enables Infrastructure as a Service (IaaS) providers such as Amazon EC2 to provide a closed box execution environment that guarantees confidential execution of guest virtual machines. Moreover, it allows users to attest to the IaaS provider and determine whether or not the service is secure before they launch their virtual machines.

**Title:** Tamper detection in audit logs

**Author:** R. T. Snodgrass, S. S. Yao, and C. Collberg

**Year:** 2004

**Description:**

Audit logs are considered good practice for business systems, and are required by federal regulations for secure systems, drug approval data, medical information disclosure, financial records, and electronic voting. Given the central role of audit logs, it is critical that they are correct and inalterable. It is not sufficient to say, \our data is correct, because we store all interactions in a separate audit log."The integrity of the audit log itself must also be guaranteed. This paper proposes mechanisms within a database management system (DBMS), based on cryptographically strong one-way hash functions, that prevent an intruder, including an auditor or an employee or even an unknown bug within the DBMS itself, from silently corrupting the audit log. We propose that the DBMS store additional information in the database to enable a separate audit log validate to examine the database along with this extra information and state conclusively whether the audit log has been compromised. We show with an implementation on a high-performance storage engine that the overhead for auditing is low and that the validate can efficiently and correctly determine if the audit log has been compromised.

**Title:** Apel: An implementation of grid accounting using r-gma

**Author:** L. C. M. C. Rob Byrom, Roney Cordenonsib

**Year:** 2005

**Description:**

This article describes the implementation of an accounting tool in the LHC Computing Grid (LCG): a distributed computing grid project consisting of over 100 resource centers and more than 10,000 CPUs. APEL (Accounting Processor for Event Logs) parses batch, system and gatekeeper logs generated by a site and builds accounting records, which provide a summary of the resources consumed based on attributes such as CPU time, Wall Clock Time, Memory and grid user DN. The accounting data is published into the R-GMA information and monitoring system, and archived for processing by a graphical front-end utilized by the accounting web tool.

**Title:** Condor-g: A computation management agent for multi-institutional grids

**Author:** Frey, Tannenbaum, Livny, Foster, and Tuecke

**Year:** 2002

**Description:**

In recent years, there has been a dramatic increase in the amount of available computing and storage resources, yet few have been able to exploit these resources in an aggregated form. We present the Condor-G system, which leverages software from Globus and Condor to allow users to harness multi-domain resources as if they all belong to one personal domain. We describe the structure of Condor-G and how it handles job management, resource selection, security and fault tolerance.

**Title:** Grasp: A grid resource allocation system based on ogsa

**Author:** O.-K. Kwon, J. Hahm, S. Kim, and J. Lee

**Year:** 2004

**Description:**

In this paper, we describe GRASP, a grid resource allocation system based on OGSA. In order to submit job to the grid resources in more efficient and convenient manner, we support some features for user-friendly resource allocation such as resource brokering, scheduling, monitoring, and so forth. GRASP supports any scientific applications with the high performance computing features such as MPI and applications with high throughput computing features such as parameter studies.

**Title:** Secure virtual machine execution under an untrusted management os

**Author:** C. Li, A. Raghunathan, and N. K. Jha

**Year:**  2010

**Description:**

Virtualization is a rapidly evolving technology that can be used to provide a range of benefits to computing systems, including improved resource utilization, software portability, and reliability. For security-critical applications, it is highly desirable to have a small trusted computing base (TCB), since it minimizes the surface of attacks that could jeopardize the security of the entire system. In traditional virtualization architectures, the TCB for an application includes not only the hardware and the virtual machine monitor (VMM), but also the whole management operating system (OS) that contains the device drivers and virtual machine (VM) management functionality. For many applications, it is not acceptable to trust this management OS, due to its large code base and abundance of vulnerabilities. In this paper, we address the problem of providing a secure execution environment on a virtualized computing platform under the assumption of an untrusted management OS. We propose a secure virtualization architecture that provides a secure run-time environment, network interface, and secondary storage for a guest VM. The proposed architecture significantly reduces the TCB of security-critical guest VMs, leading to improved security in an untrusted management environment. We have implemented a prototype of the proposed approach using the Xen virtualization system, and demonstrated how it can be used to facilitate secure remote computing services. We evaluate the performance penalties incurred by the proposed architecture, and demonstrate that the penalties are minimal.

**Title:** vtpm: virtualizing the trusted platform module

**Author:** S. Berger, R. C´aceres, K. A. Goldman, R. Perez, R. Sailer, and L. van Doorn

**Year:**  2006

**Description:**

We present the design and implementation of a system that enables trusted computing for an unlimited number of virtual machines on a single hardware platform. To this end, we virtualized the Trusted Platform Module (TPM). As a result, the TPM’s secure storage and cryptographic functions are available to operating systems and applications running in virtual machines. Our new facility supports higher-level services for establishing trust in virtualized environments, for example remote attestation of software integrity. We implemented the full TPM specification in software and added functions to create and destroy virtual TPM instances. We integrated our software TPM into a hypervisor environment to make TPM functions available to virtual machines. Our virtual TPM supports suspend and resume operations, as well as migration of a virtual TPM instance with its respective virtual machine across platforms. We present four designs for certificate chains to link the virtual TPM to a hardware TPM, with security vs. efficiency trade-offs based on threat models. Finally, we demonstrate a working system by layering an existing integrity measurement application on top of our virtual TPM facility.

**Title:** Tisa: Toward trustworthy services in a service-oriented architecture

**Author:** H. Rajan and M. Hosamani

**Year:**  2008

**Description:**

Verifying whether a service implementation is conforming to its service-level agreements is important to inspire confidence in services in a service-oriented architecture (SoA). Functional agreements can be checked by observing the published interface of the service, but other agreements that are more non-functional in nature, are often verified by deploying a monitor that observes the execution of the service implementation. A problem is that such a monitor must execute in an untrusted environment. Thus, integrity of the results reported by such a monitor crucially depends on its integrity. We contribute an extension of the traditional SoA, based on hardware-based root of trust, that allows clients, brokers and providers to negotiate and validate the integrity of a requirements monitor executing in an untrusted environment. We make two basic claims: first, that it is feasible to realize our approach using existing hardware and software solutions, and second, that integrity verification can be done at a relatively small overhead. To evaluate feasibility, we have realized our approach using current software and hardware solutions. To measure overhead, we have conducted a case study using a collection of Web service implementations available with Apache Axis implementation.

**Title:** Inspector gadget: a framework for custom monitoring and debugging of distributed data flows

**Author:** C. Olston and B. Reed

**Year:**  2011

**Description:**

We consider how to monitor and debug query processing data flows, in distributed environments such as Pig/Hadoop. Our work is motivated by a series of informal user interviews, which revealed that monitoring and debugging needs are both pressing and diverse. In response to these interviews, we created a framework for custom dataflow instrumentation, called Inspector Gadget (IG). IG makes it easy to write a wide variety of monitoring and debugging behaviors, and attaches seamlessly to an existing, unmodified dataflow environment such as Pig. We have implemented a dozen user-requested tools in Inspector Gadget, each in just a few hundred lines of Java code. The performance overhead is modest in most cases. Our Pig-based implementation of IG, called Penny, is slated for public release in mid-2011, in conjunction with the upcoming Apache Pig v0.9 release.

**Modules:**

* User Interface Design
* Cloud Service Provider
* User
* Cloud Notary Authority (CNA)
* Monitor
* Action against SLA violation.

**Modules Description:**

**User Interface Design**

User Interface Design have a purpose that a user to move from login page to user page of the website. In this we want to enter our user name and password provided by Service provider. If we enter the valid password and user name then only the user can move login page to user window while entering user name and password it will check username and password is match or not. If we enter any wrong username or wrong password it generates some error message. So we are preventing from unauthorized user entering into the service provider website. It will provide a good security for our project. So Service provider contain user name and password server also check the authentication of the user. It will improve the security and preventing from unauthorized user enters into the website. In our project we are using java swings for creating design. Here we are validating the users who are going to access the Service providers.

**Cloud** **Service Provider**

Service provider has a job of providing a service like software to the cloud users. In our proposed method, CSP doesn’t provide billing transaction to the user. It is due to the reason if billing transaction performed in the CSP then complexity in security to be provided for billing transaction increases the overhead. If the user logged in for service, CSP validate the user whether he\she is an authenticated user or not. Once if user is found authenticated user then it waits for service check in message else it found any unauthenticated user it will send the error message. If it received the service check in message then it responds the user by transmitting the agreement and hash chain (one time key). After getting the service request from the user, CSP provide the requested service to the user. It is also have a contact with the Cloud notary authority. It will provide the service until it receive the service checkout message. The CSP enables users to scale their capacity upwards or downwards regarding their computing requirements and to pay only for the capacity that they actually use.

**User**

User can access a service from the Cloud Service Provider by authenticated login process. We assume that users are thin clients who use services in the cloud computing environment. To start a service session in such an environment, each user makes a service check-in request to the CSP with a billing transaction. To end the service session, the user can make a service check-out request to the CSP with a billing transaction. Once if the users send the service check-in message it can get the contract from the CSP. After receiving the one time keywords in the contract it can be able to access the service from the CSP. Now user log details are stored in Monitor for future disputes. After accessing the service, user want billing transaction. If he\she wants the bill means it should send the contract of the CSP with contract of the user to the CNA. If both the details checked by the CNA are identical then user can receive the bill binding information along with confirmation message. If any error occurred or forgery activity found from the user side then the user will receive the penalty for that.

**Cloud Notary Authority (CNA)**

Cloud Notary Authority acts as a THEMIS in our cloud billing transaction. He is an authority to generate the billing transaction for the cloud service. The CNA provides a mutually verifiable integrity mechanism that combats the malicious behavior of users or the CSP. The process, which involves a generation of mutually verifiable binding information among all the involved entities on the basis of a one-way hash chain (One time key), is computationally efficient for a user and the CSP. If user wants billing for the service then it sends the contract of the user and contract of CSP to the CNA. In CNA it checks both the contract; if it is found as identical then it generates the bill as binding information and sends the confirmation message to the user and the CSP. If it is not identical then it receives the log details from the monitor. If forgery found at user side it sends the penalty to the user. If it found at CSP side it cancels the payment to the CSP. CNA provide the billing transaction which can be verifiable and also forgery resistive in cloud environment.

**Monitor**

Monitor is a module which continuously monitors all the log activities of the CSP and the user. For monitoring it uses a technique called S-Mon. The S-Mon has a forgery-resistive SLA measuring and logging mechanism, which enables it to monitor SLA violations and take corrective actions in a trusted manner. After the service session is finished, the data logged by S-Mon are delivered to the CNA. We devised S-Mon in such a way that it can be deployed as an SLA monitoring module in the computing resources of the user. Once SLA has been violated S-Mon sends all the log details to the CNA. After verifying the log details CNA perform further action. Monitor has a local repository for storing all the log details of the user to monitor the SLA for the future disputes. So it can be verifiable in future too. Here monitor plays important role against billing transaction forgery which leads to forgery resistive billing transaction.

**Action against SLA violation**

Once the CNA found forgery from cloud services it can’t directly take any action against them without knowing the reason. At that time it sends the message to Monitor to send the all log details about the transaction. Once it receives the log message from the monitor it compares the contract and the log details. Once the forgery found from CSP side it cancels the payment to the CSP and send the message to the CSP. If it found from the user side it assign penalty to the user according to the severity of the forgery from the user side and sends the message to the user. CNA also maintains the local repository after the action taken against the SLA violation.

**Module Diagrams:**

**User Interface Design**

USER

Cloud Service Provider

LOGIN

Client Account page

Validation

Database

**Cloud** **Service Provider**

**CSP**

USER

Check in

Contract + Hash chain

Software services

User details

Service details

Database

**User**

**USER**

CSP Details

CNA Details

Service logs

SLA Detail

**CSP**

CHECK IN

**CNA**

Contract of CSP + USER

**Cloud Notary Authority (CNA)**

**CNA**

CSP Details

User Details

Bill Generation

**USER**

**CSP**

**Local Repository**

Contract

BILL Transaction

BILL Transaction

**Monitor**

**MONITOR**

CSP Details

USER Details

LOG Details

**CSP**

**USER**

**Action against SLA violation**

**USER**

**CSP**

**CNA**

**MONITOR**

Log Details

Error contract

Penalty

Payment Cancel

**GIVEN INPUT EXPECTED OUTPUT**

**User Interface Design**

Input:

* + - Registration
    - Username
    - Password

Output:

* + - Registered in Database
    - Login successfully
    - Open client home page

**Cloud** **Service Provider**

Input:

* + - Check-in message
    - User response with one time key

Output:

* + - Contract including SLA and Hash chain
    - Software Service

**User**

Input:

* + - Contract and Hash chain from CSP
    - Access software from CSP

Output:

* + - Response to csp with Hash chain
    - Send contract of user and CSP to CNA

**Cloud Notary Authority (CNA)**

Input:

* + - contract of user and CSP from user

Output:

* + - Generation of billing transaction
    - Send confirmation message to CSP and user

**Monitor**

Input:

* + - Logging details of CSP
    - Logging details of user

Output:

* + - Stored in repository
    - Send logging details to the CNA when error occurred in contract

**Action against SLA violation**

Input:

* + - Logging details of user and CSP from monitor

Output:

* + - Cancel payment for CSP
    - Provide penalty for user
    - Maintain local repository for future dispute

**Technique description:**

**CLOUD NOTARY AUTHORITY (CNA)**

The CNA provides a mutually verifiable integrity mechanism that combats the malicious behavior of users or the CSP. The process, which involves a generation of mutually verifiable binding information among all the involved entities on the basis of a one-way hash chain, is computationally efficient for a thin client and the CSP.

**SLA MONITOR USING SMon**

The S-Mon has a forgery-resistive SLA measuring and logging mechanism, which enables it to monitor SLA violations and take corrective actions in a trusted manner. After the service session is finished, the data logged by S-Mon are delivered to the CNA. We devised S-Mon in such a way that it can be deployed as an SLA monitoring module in the computing resources of the user.

**SYSTEM REQUIREMENTS**

**HARDWARE**

PROCESSOR : PENTIUM IV 2.6 GHz, Intel Core 2 Duo.

RAM : 512 MB DD RAM

MONITOR : 15” COLOR

HARD DISK : 40 GB

**SOFTWARE**

Front End : J2EE (JSP, SERVLET)

Back End : MS SQL 05

Operating System : Windows 07

IDE : Net Beans, Eclipse

**SYSTEM DESIGN**

**Use Case Diagram:**



**EXPLANATION:**

From the above use case diagram we can know that every actor have some action with other actor in the network. User have log in relation with CSP. For that CSP response with contract contains one time key. For billing transaction USER send the contract of both user and CSP to the CNA. Then CNA produce Billing transaction and also send the confirmation message to both user and CSP. All the logging details are stored by monitor which will be send to the CNA for disputes.

**CLASS DIAGRAM:**



**EXPLANATION:**

From this class diagram how the classes are interconnected to perform the action is explained. Service provider class is contact with the user class for validating the user and sends the contract. After that user class have send the request to the CNA class for bill generation and it calls the both service provider and user class for confirmation. After finishing it receives message from monitor for checking log details.

**OBJECT DIAGRAM:**



**EXPLANATION:**

From the object diagram we can know that how we create a object for a class and how they perform the action. Simply it shows that flow of object. We create object for a class like service provider, authority, user and monitor to perform the action of bill generation in efficient way. It is a diagram that shows a complete or partial view of the structure of a modeled system.

**STATE DIAGRAM**



**EXPLANATION:**

In this State diagram the process starts from validation of the user and then proceed with sending the contract. After accessing the service user send the contract to the authority. By checking the both contract authority will generate the bill. If it is error it checks the log details from the monitor and produce penalty or payment cancellation. After that it can send either bill or error message to botrh user and csp at that point process come to end.

**ACTIVITY DIAGRAM**



**EXPLANATION:**

In this activity diagram user can access service from the CSP by sending the check in message. After accessing the service if user wants billing transaction it sends the contract of the user and CSP to authority. Authority checks both contract if it is identical then it generates the bill and send to both the user and csp. If it found any mismatch then it checks the log details from the monitor and sends the action against the CSP/user.

**Sequence Diagram**

****

**EXPLANATION:**

From this sequence diagram it is shown that the sequence from user to can for bill generation. First user get a contract by check in message and access the service using that contract. After accessing the service it request the CNA for bill generation. CNA checks the both contract if it is identical then it generates the bill and send the information to the user and csp. If it is not identical then it checks the log details from monitor and take action against the SLA violator.

**Collaboration Diagram**



**EXPLANATION:**

Collaboration diagram shows how the Can takes the dyanamic action like if it founds the contract are identical then it generates the bill or else it checks the log details from the moniotor and take the action against the violator. So this type of dynamic flow and astatic flow like user validation and response activity are represented by using this diagram.

**COMPONENT DIAGRAM**



**EXPLANATION:**

From this diagram components like cna, monitor, user, csp are connected to form the larger components. So can have further decision making dynamically. If it receives identical contract then it generates bill otherwise it check the log details from monitor and take further decision depends on where the violation in SLA takes place.

**DATA FLOW DIAGRAM**

**Level 0:**

USER

CSP

MONITOR

**Level 1:**

CNA

USER

CSP

MONITOR

**EXPLANATION:**

From DFD 0 the flow from user to csp is shown. User requests the CSP and gets the service from the CSP after giving confirmation with hash chain. Monitor has the log details of both monitor and the CSP.

From DFD 1 we can know that after accessing the service from CSP, user sends the contract of user and CSP to the CNA. CNA Verify both the contracts if both are identical then it generates the bill and sends to both user and CSP. If not it checks the log details at the monitor and take further action against the violator.

**E-R DIAGRAM**

CNA

USER

CSP

MONITOR

Contract

BILL

Validation

Log

Log

Check log

Bill transaction

**EXPLANATION:**

E-R diagram shows that relationship between the various entities with attributes. In this diagram user have attributes like username and password having validation and service relationship with the entity CSP with the attributes contract, hash chain and software. And also logging relationship with entity monitor. Likewise every entity have a relationship with other entities.

**System Architecture:**

Validation

Contract

User log

Contract of User + CSP

ERROR in contract

USER

CSP

CNA

MONITOR

Cancel payment

Penalty

DATABASE

DATABASE

CSP Log

**EXPLANATION:**

System Architecture shows that the entire flow of the project. When the users are validated by the CSP, it will send the contract with hash chain to the user. After that the user request for the service with that hash key. Once user finished accessing service from the CSP it sends the contract of the user and CSP to the authority. Authority checks the contract; if it is identical then it generates the bill and send the confirmation message to the CSP and the user. If it found error it checks the log detail from the monitor and take the action against the person who violates the service level agreement.

**FUTURE ENHANCEMENT**

In future, the deployment of THEMIS in the context of existing cloud computing services requires minimal modification to the CSPs, CNA and users if seeking to provide mutually verifiable billing transactions. Our next step is to consider the scalability and fault tolerance of THEMIS. We believe that putting multiple trusted third parties in charge of the CNA is an appropriate way forward, as is the case with the PKI. We are working towards a THEMIS-based system with more fault tolerance against scalable billing.

**ADVANTAGES**

* Billing transactions are non obstructive.
* Minimal Computation Cost.
* Trusted Service level agreement (SLA) monitoring by SMon.
* Minimum transaction latency.

**APPLICATIONS**

**ePN Mobile iPhone**

This mobile phone have a application of transaction processing available at swiped rates through common smart phones, cell phones and PDA's. The ePN Mobile Credit Card, Check and Gift/Loyalty Application can prompt for invoice number, gratuity, other charges process the transaction real-time and show the transaction authorization number right on the phone display.

**VOSS Fulfillment Solution**

Specialty OSS vendors (Operational Support Systems) have developed end-to-end service orchestration solutions for service providers in the cloud communications space. VOSS Solutions is the leading OSS vendor in this public, cloud communications OSS space, with more tier-1 service provider customers than any other player.

**Absolute Performance SLA Monitoring**

Organizations have an increasing demand for business visibility.  As a business executive, it is vital to know the state of your business-critical and revenue critical applications at all times.  Knowing that your application is being managed to meet your business requirements is necessary to ensure 24x7 availability, transaction volume and performance of the application from the end-user perspective.  Absolute Performance provides the visibility through custom SLA monitoring, enabling executives to view real-time SLA compliance and reporting, consolidated into a cohesive, easy to use portal view.

**CONCLUSION**

THEMIS significantly reduces the billing transaction overhead. It provides a high securable and non obstructive billing system. Central Nodal Authority (CNA) generates the bill with binding information. It acts as forgery-resistive SLA measuring and logging mechanism. So even administrator of a cloud system cannot modify or falsify the data.

**REFERENCE OR BIBLIOGRAPHY**

1. L. C. M. C. Rob Byrom, Roney Cordenonsib, “Apel: An implementation of grid accounting using r-gma,” UK e-Science All Hands Conference, Nottingham, September 2005.

2. Frey, Tannenbaum, Livny, Foster, and Tuecke, “Condor-g: A computation management agent for multi-institutional grids,” Cluster Computing, vol. 5, pp. 237–246, 2002.

3. O.-K. Kwon, J. Hahm, S. Kim, and J. Lee, “Grasp: A grid resource allocation system based on ogsa,” in Proc. of the 13th IEEE Intl. Sympo. on High Performance Distributed Computing. IEEE

4. Computer Society, 2004, pp. 278–279. I. P. Release, “Tivoli: Usage and accounting manager,” IBM Press, 2009.

5. H. Rajan and M. Hosamani, “Tisa: Toward trustworthy services in a service-oriented architecture,” IEEE Transactions on Services Computing, vol. 1, pp. 201–213, 2008.

6. S. Meng, L. Liu, and T. Wang, “State monitoring in cloud datacenters,” IEEE Transactions on Knowledge and Data Engineering, vol. 23, pp. 1328–1344, 2011.

7. C. Olston and B. Reed, “Inspector gadget: a framework for custom monitoring and debugging of distributed dataflows,” in Proc. of the 2011 international conference on Management of data,

ser. SIGMOD ’11. ACM, 2011, pp. 1221–1224.

8. P. Leitner, A. Michlmayr, F. Rosenberg, and S. Dustdar, “Monitoring, prediction and prevention of sla violations in composite services,” in Proc. of the 2010 IEEE ICWS. IEEE Computer Society, 2010, pp. 369–376.

9. S. Pearson and B. Balacheff, Trusted computing platforms: TCPA technology in context, ser. HP Professional Series. Prentice Hall PTR, 2003.

10. I. P. Release, “White paper: Trusted execution technology, hardware-based technology for enhancing server platform security,” Intel Press, Tech. Rep., 2010.