INTERNATIONAL JOURNAL OF ENHANCED RESEARCH IN MANAGEMENT & COMPUTER APPLICATIONS
VOL. 2, ISSUE 3, MARCH. 2013 ISSN NO: 2319-7471

A THEMIS Billing System for the Cloud Computing Environment

Thoudam Johnson Singh[1], Dr. R. Sugumar[2],
[1]M.E. Student – VelTech Multitech Dr. Rangarajan Dr. Sakunthala Engg College
[2]Associate Professor - VelTech Multitech Dr. Rangarajan Dr. Sakunthala Engg College
[1]jonthoudam@gmail.com, [2]sugu16@gmail.com

Abstract: With the advent and acceptance of cloud computing the ability to record and account for the usage of cloud resources in a convincing and capricious way has become critical for cloud service providers and users alike. The billing process involves receiving billing records from various networks, determining the billing rates associated with the billing records, calculate the cost for each billing record, aggregating this records periodically to generate invoices, sending invoices to the customer, and collecting payments received from the customer. Traditional billing systems are not enough in terms of security capabilities or computational overhead. In this paper, we propose a secure and no obstructive billing system called THEMIS as a remedy for these limitations. The system uses a novel concept of a cloud notary authority for the supervision of billing. Moreover, to provide a falsification-resistive SLA monitoring mechanism, we devised a SLA monitoring module enhanced with a trusted platform module (TPM), called S-Mon. From the perspective of extensibility, THEMIS is moreover applicable to various target services as well to improve the accountability of each service by applying more monitoring techniques to S-Mon. The overall latency of these systems is shorter and the throughput much higher than the Public key Infrastructure.

Keywords: Records, verification, transaction processing, pricing and resource allocation.

I. INTRODUCTION

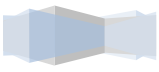
Cloud computing is representative of the most important transition and paradigm shift in service-oriented computing technology. Emerging cloud services, such as Amazon EC2, S3 [1], and Microsoft Azure[2], have be- come popular in recent years. Although cloud computing has its roots in grid computing and utility computing technologies, it differs significantly from those technologies in terms of its service model.

Cloud service providers (CSPs) generally use a pay- per-use billing scheme in their pay-as-you-go pricing model: that is, the consumer uses as many resources as needed and is billed by the provider for the amount of resources consumed by the end of an agreed-upon period. CSPs usually guarantee the quality of service (in terms of availability and performance) in the form of a service level agreement (SLA) [3]. An SLA is supported by clear metrics and regular performance monitoring. In this service model, users who use an infrastructure-as-a- service (IaaS) may wish to figure out the billed charges for the total service time and the guaranteed service level. If a company uses a platform-as-a-service (PaaS) or software-as-a-service (SaaS), the accounting department of the company may require the service usage logs so as to verify the billed charges by checking the company's total number of running software programs or platforms. We refer to this type of transaction as a billing trans- action; it is used to keep track of cloud service usage records and to verify whether the CSP has offered the quality of service promised under the SLA arrangement.

Providing a billing mechanism in a trusted manner is critical for CSPs and users[4]. However, the security aspects of a cloud billing system and the scale of cloud services often raise the following security and system issues:

A. Billing transaction with integrity and nonrepudiation capabilities

For transparent billing of the cloud services, each billing transaction should be protected against forgery and false modifications [5]. Although commercial CSPs [1], [2] provide users with service billing records and while several researchers have presented resource usage processing systems [6], [7], [8], [9] that record the use of grid resources, they cannot provide a trustworthy audit trail. It is because the user or the CSP can modify the billing records even after a mutual agreement between the user and the CSP, leading to the dispute between them. In this case, even a third party cannot confirm that the



user 's record is correct or that the CSP's record is correct. Therefore, a trustworthy audit trail is important for resolving disputes, and the billing record in the billing transaction must be assuredly incorruptible per mu- tual agreement. One way of ensuring the integrity and nonrepudiation of a transaction (that is, where partici- pants cannot deny the context of a billing transaction) is to integrate a public key infrastructure (PKI)-based digital signature into each billing transaction to prevent corruption. Several studies have addressed this issue by deploying a PKI-based digital signature mechanism in an underlying security layer; however, they were handicapped by computational overhead due to the extreme complexity of the PKI operations.

B. State Monitoring

Despite the distributed nature of cloud-hosted applications, application owners often need to monitor the global state of deployed applications for various purposes. For instance, Amazon's Cloud Watch [8] enables users to monitor the overall request rate on a web application deployed over multiple server instances. Users can receive a state alert when the overall request rate exceeds a threshold, e.g., the capacity limit of provisioned server instances. In this case, users can deploy the web application on more server instances to increase throughput.

State monitoring in data centers poses two fundamental requirements. First, given the serious outcome of incorrect monitoring results, state monitoring must deliver correct monitoring results [11]. A false state alert in the previous Cloud Watch example would cause provisioning of new server instances which is clearly unnecessary and expensive. Missing a state alert is even worse as the application gets overloaded without new server instances, which eventually causes potential customers to give up the application due to poor performance. This correctness requirement still holds even if monitored values contain momentary bursts and outliers.

Second, communication related to state monitoring should be as little as possible [4], [5], [6]. Data centers usually run a large number of state monitoring tasks for application and infrastructure management [1]. As monitoring communication consumes both bandwidth and considerable CPU cycles [4], state monitoring should minimize communication. This is especially important for infrastructure services such as EC2, as computing resources directly generate revenues.

C. The Window-Based State Monitoring

As monitored values often contain momentary bursts andoutliers, instantaneous state monitoring [16] is subject to cause frequent and unnecessary state alerts, which could further lead to unnecessary countermeasures. Since short periods of state violation are often well acceptable, a more practical monitoring model should tolerate momentary state violation and capture only continuous one. Therefore, we introduce window-based state monitoring which triggers state alerts only when the normal state is continuously violated for L time units. We study window-based state monitoring instead of other possible forms of state monitoring for two reasons. First, we believe continuous violation is the fundamental sign of established abnormality. Second, window-based state monitoring tasks are easy to configure, because the window size L is essentially the tolerable time of abnormal state, e.g., degraded service quality, which is known to service providers.

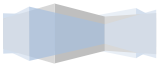
D. WISE Monitoring Approach

We now focus on the three technical developments that form the core of the WISE monitoring approach: the WISE monitoring algorithm, the monitoring parameter tuning schemes, and performance optimization techniques. The right side of Fig. 1 shows a high level view of the WISE monitoring approach, Fig. 2. shows WISE monitoring system. 3.2.1 The Monitoring Algorithm The idea behind the WISE monitoring algorithm is to report partial information on local violation series at the monitor node side to save communication cost. The coordinator collects further information only when the possibility of detecting state alerts cannot be ruled out.

Specifically, the monitor node side algorithm employs two monitoring parameters, the local threshold Ti and the filtering window size pi. When detects local violation (viðtÞ > Ti), a monitor node i sends a local violation report and starts a filtering window with size pi during which it only records monitored values and does not send violation reports.

II. RELATED WORKS

Billing systems that track and verify the usage of computing resources have been actively studied and developed in the research area of grid and cloud computing. Many studies have analyzed pre-existing billing systems of grid and cloud computing environments. They have tried to identify the new requirements of the shift in the computing paradigm from grid computing to cloud computing. In this section, we briefly discuss experimental results as we evaluate existing billing systems in terms of their security level and billing overhead. We evaluate the billing systems in an identical computing and network environment.



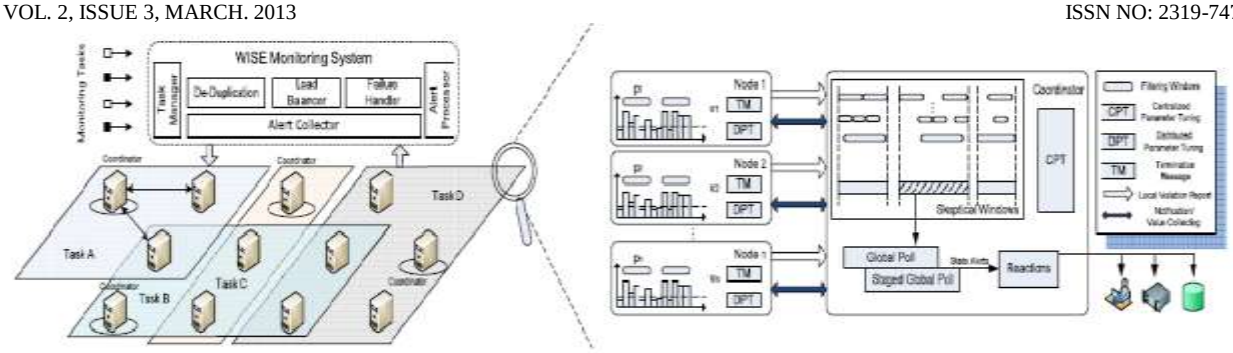


Fig. 1 Wise Monitoring System

A. Billing Systems with Limited Security Concerns

Two pioneering studies identified challenges in man- aging the resources of a grid computing environment and proposed a computational economy as a metaphor for effective management of resources. The resource usage information, which pertains to the CPU cycles, storage, and network bandwidth, is collected via a resource usage monitor and charged over the billing agent. APEL [6] presents a billing system that processes log information to create quantified accounting records. Other resource management and billing frameworks that were suggested as part of traditional grid approaches: namely, Condor/G [7], GRASP [8], and Tivoli [9].

B. Security Enhanced billing system

Several electronic payment schemes have been pro- posed in the literature in an attempt to provide security-enhanced billing mechanisms. As shown in Fig. 1b, the micropayment-based scheme has a short billing latency (4.70 ms) in our experimental environment. However, it cannot support the security features of nonrepudiation and trusted SLA monitoring because micropayment schemes are mainly designed for transaction integrity rather than security features.

C. PKI Based Billing System

The organization of a PKI-based billing system and its characteristics in terms of the security level and billing overhead. It has a longer billing latency (82.51ms) than the other systems in our experimental environment. The extent of the overhead is mainly determined by the extremely high complexity of the RSA [31] operations when the PKI is used for a billing system by a thin client or a heavily loaded server. The computational overhead can be a severe drawback when a number of cloud service users and the CSP generate a vast amount of billing transactions.

III. DESIGN OF THEMIS BILLING SYSTEM

We present an overview of the THEMIS billing system in this section. We first introduce the important components of THEMIS and describe the overall billing process.

A. The Proposed Themis Infrastucture

Fig. 3 shows the overall architecture of THEMIS billing system. The four major components of the architecture are listed as follows:

• Cloud Service Provider (CSP): The CSP nables users to scale their capacity upwards or downwards regarding their computing requirements and to pay only for the capacity that they actually use.

• Users: 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.

• 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.

• Trusted SLA Monitor (S-Mon): The S-Mon has a forgery-resistive SLA measuring and logging mech- anism, 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.

B. Overall Billing Process of THEMIS

After a registration phase, THEMIS can use the above components to provide a mutually verifiable billing transaction without asymmetric key operations of any entities. The registration phase involves mutual authen- tication of the entities and the generation of a hash chain by each entity. The hash chain element of each entity is integrated into each billing transaction on a chain-by-chain basis; it enables the CNA to verify the correctness of the billing transaction. In addition, S- Mon has a forgery-resistive SLA measuring and log- ging mechanism. THEMIS consequently supervises the billing; and, because of its objectivity, it is likely to be accepted by users and CSPs alike.

The billing transactions can be performed in two types of transactions: a service check-in for starting a cloud service session and a service check-out for finalizing the service session. These two transactions can be made in a similar way. Each billing transaction is performed by the transmission of a message, called a μ-contract. A μ- contract is a data structure that contains a hashed value of a billing context and the hash chain element of each entity. With the sole authority to decrypt both the μ-contract from the CSP and the μ-contract of the user, the CNA can act as a third party to verify the consistency of the billing context between the user and the CSP.

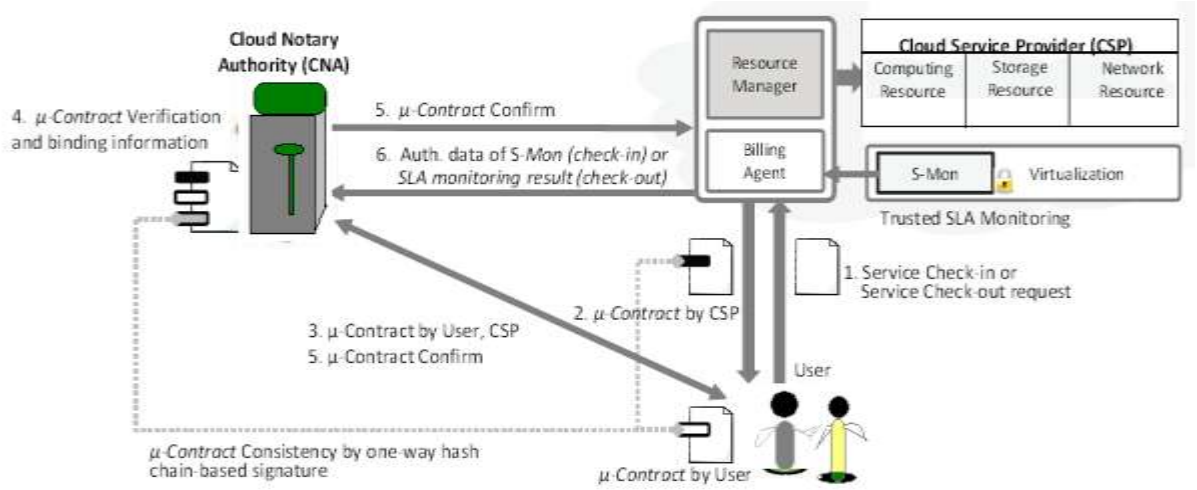


Fig. 3. Overall architecture and flowchart of the billing transactions of THEMIS

Fig. 3 shows the overall process of billing transaction with our billing system. The main steps are as follow:

1) The user generates a service check-in or check-out request message and sends it to the CSP.
2) The CSP uses an element from the CSP's hash chain to send the user a μ-contract-CSP as a digital signature.
3) The user uses an element from the user 's hash chain to generate a μ-contract-User as a digital signature. The user then combines the μ-contract- User with μ-contract-CSP and sends the combined μ-contract to the CNA.
4) 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.
5) The billing process is completed when the user and the CSP receive confirmation from the CNA.
6) Finally, in the case of a service check-in, the S-Mon of the user 's cloud resource transmits authentica- tion data of the S-Mon to the CNA. In the case of a service check-out, S-Mon sends a report of the SLA monitoring results to the CNA.
A more descriptions of above transactions can be found in section 4.

IV. Proposed Billing Protocol

In this section, we describe the overall transactions of the proposed billing scheme; and, in Section 4.4, we analyze the security and safety of the proposed billing system.

A. Description of the THEMIS Billing Protocol

Fig. 4 shows a flow diagram of the overall transactions of the proposed billing protocol. The protocol consists of the three states.
.
-State 1 (Mutual Authentication): This state is for a user who accesses the CSP for the first time. When the user first accesses the CSP, PKI-based authentications are per- formed by the user, the CSP, and the CNA. Throughout the mutual authentications, the user, the CNA, and the CSP exclusively share the following three keys:
• CSP ↔ CNA: Kc,n
• User ↔ CNA: Ku,n
• User ↔ CSP: Ku,c

-State 2 (Hash Chain Generation): This state is for generating and registering a hash chain among the CSP, the CNA, and the user. Each of these three parties generates a hash chain of length N by applying the hash function N times to a seed value (Cu,N , Cc,N , and Cn,N) so that a final hash (Cu,0 , Cc,0 , and Cn,0) can be obtained. As shown in Fig. 5, the user and the CSP commit to the final hash by digitally signing the final hash (Cu,0 and Cc,0), and by registering the signed hash chain elements
to the CNA.
-State 3 (Billing Transaction): An actual billing transac- tion is performed at the beginning of State 3. In this state, a user can perform two types of billing transactions: 'a service check-in' (to start a service session, State 3-1) and'a service check-out' (to end a service session, State 3-2). The service check-in is for requesting a new cloud service, such as a virtual machine service or a storage service. A user who wants to end the cloud service can perform a billing transaction for 'a service check-out'.
Both types of transactions can be performed in a similar way. The difference between them is the context of the message. The context of a service check-in includes the SLA and data for the initialization of SLA monitoring. The context of the service check-out includes information that can be used to verify data from the SLA monitoring module, S-Mon.

• State 3-1 (Billing transaction for a service check- in): As shown in Fig. 5, a user who intends to receive a cloud service from a CSP sends a service check- in request message (Message 3-1) to the CSP. Upon receiving the message, the CSP transmits a stipulation (S) and a μ-contract-CSP to the user. The S contains a service invoice and an SLA that covers guaranteed performance factors, such as availability, CPU speed, I/O throughput, a time stamp, and the price. The μ- contract-CSP contains a hash chain element of the CSP, Cc,n . The hash chain element, which is listed in Table 1, is updated for each μ-contract-CSP on a chain-by-chain basis so that all of the μ-contract-CSP can be linked and verified sequentially toward the seed value (Cc,0) of the hash chain.
• State 3-2 (Billing transaction for a service check- out): A user who intends to end the cloud service of the CSP performs a billing transaction that is similar to the service check-in transaction. The main difference between the check-in and check-out transactions is that S-Mon sends an SLA monitoring result to the CNA in the confirmation message (Message 3-5). The CNA can consequently determine whether the SLA has been violated. If the CSP is unable to meet the SLA, the CNA may impose penalties, such as reducing or canceling the payment.

B. S-Mon: SLA Monitor

To provide trusted SLA monitoring, we devised S-Mon which can be deployed into computing resources of the CSP. S-Mon provides a forgery-resistive SLA measuring and logging mechanism in a black-box manner. Thus, even the administrator of the CSP cannot modify or falsify the logged data.
The S-Mon procedure consists of three phases. Phase1 is performed for the beginning of a service session; Phase2 is periodically invoked during a service session; and Phase3 is performed for the end of the service session. The CNA can consequently determine whether the SLA has been violated. We note that S-Mon deliver the logged data to the CNA only after the service session is finished or when the user requires SLA monitoring data from the CNA via Phase3. The details of the three S-Mon phases are as follows:

• Phase1 (S-Mon Initialization): In a billing transaction for a service check-in, the S-Mon of the user 's cloud resources initializes itself by accepting S. To enable the CNA to check the freshness of the S-Mon, S-Mon performs an extend() operation by inputting H (S) and a tick-stamp (line 2). The tick-stamp confirms the starting time of S-Mon.

For secure communication between S-Mon and the CNA, S-Mon generates a pair of private and public keys, SKm and P Km (line 3). The counter of line 5 is the value of the monotonic counter of S-Mon's TPM. The counter and Seed (initial value=Nu) are used in Phase2 and Phase3 for the integrity check of the data logged by S-Mon.

• Phase2 (SLA Monitoring): Phase2 periodically occurs during the service time. To ensure its own execution integrity, S-Mon performs this phase only when the counter value stored in the NV is the same as the monotonic counter value of the TPM (line 1-3). Whenever this phase occurs, the monotonic counter is increased, and the value of the counter is stored in the NV (line7). Thus, when Phase2 is executed again, the counter value in the NV must be the same as the value of the monotonic counter. Otherwise S-Mon is aborted, which means the counter value was tampered with or the unsealed data were stale.

• Phase3 (SLA Report): Phase3 is executed when the corresponding service session is ended by the user. S- Mon transmits the BB which contains the SLA monitor- ing result to the CNA. Before sending the BB, S-Mon appends the final tick-stamp, the counter, and Seed (line

3). S-Mon stores the current status itself by using the Extend() operation (line 4). A digital signature of TPM is used to bind BB, Nu , and Auth to q with the PCR values (line 6). Finally, S-Mon returns the results with its digital signature by SKm to the CNA (line 7). The context of BB enables the CNA to check whether S-Mon was executed correctly without a break or halt and whether the returned result was truly generated by S-Mon.

V. PERFORMANCE EVALUATION

In this section, we present the performance results of our prototype version of THEMIS. First, we demonstrate the overall experimental environment. We then describe the operational efficiency of the billing protocol to eval- uate the performance of THEMIS in terms of latency and throughput. Finally, we present the performance.

A. Throughput evaluation

Fig. 4 illustrates how the throughput of the billing transactions mutates as the number of billing requests per second varies. The number of billing requests per second ranges from 1,000 to 15,000. For the PKI-based billing protocol, we found that the throughput was satu- rated on 903.3 transactions per second as the number of billing requests increased. This outcome is due mainly to the cryptography operations and the communication overhead of both the client side and the server side.

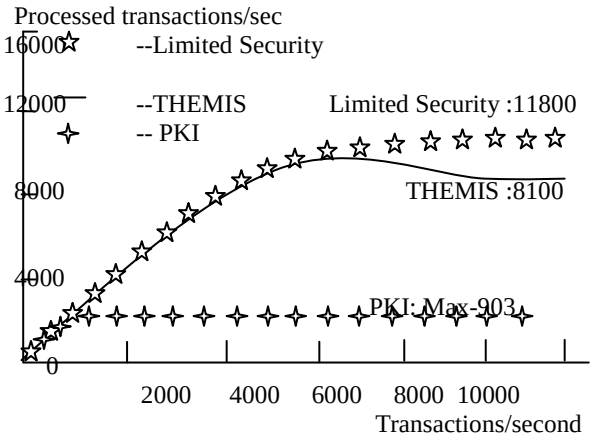


Fig 4: Throughputs of the billing transactions with varying number of billing transactions per second.

In the case of the billing system with limited security concerns as described before, the throughput is saturated on 11800 transactions per second. This outcome is due to their lower computation and commu- nication overhead than the other systems. In the case of THEMIS the throughput is saturated on 8100 transactions per second, respectively, as the number of billing requests increases. This phenomenon is due to the fact that the quantity of THEMIS and micropayment operations of the user and server provider is much smaller than that of PKI-based billing. This result confirms that the THEMIS billing protocol can seamlessly provide a nonobstructive billing transaction whenever the number of requests per second is less than 9000. From the perspective of performance saturation, we believe that putting multiple trusted third parties in charge of the cloud notary authority is an appropriate way forward, as is the case with the PKI. We are working towards a THEMIS-based system with more fault tolerance to scalable billing.

CONCLUSION

The increasing use of consolidation and virtualization is driving the development of innovative technologies for managing cloud applications and services. We argue that THEMIS is one of the crucial functionalities for on demand provisioning of resources and services in cloud datacenters. We have presented a distributed approach for efficient and mutually verifiable billing system for cloud environment that improves the accountability as well. To derive these approaches we've studied the various existing billing system in the environment. We conceived and implemented the concepts of a CNA and S-Mon in our proposed system that helps to supervise and implement the system in a more objective and acceptable way.
Our system has three features that the existing systems has not : First, a concept of CNA to give an undeniable transaction between the clients and users. Second, mutually verifiable billing protocol that replaces the existing PKI based operations and, Finally, deployment of forgery registive SLA measuring and logging mechanism.

REFERENCES

[1]. A. C. Ltd., "Amazon elastic compute cloud ec2, simple storage service," Amazon, http://aws.amazon.com/ec2/, http://aws.amazon.com/s32/, April 2011.
[2]. Microsoft, "Microsoft, windows azure platform," 2010.[Online].Available:http://www.microsoft.com/windowsazure.
[3]. 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.
[4]. 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.
[5]. 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.
[6]. 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.
[7]. K.-W. Park, S. K. Park, J. Han, and K. H. Park, "Themis: Towards mutually verifiable billing transactions in the cloud computing environment," in Proc. of the IEEE 3rd Intl. Conf. on Cloud Computing, 2010, pp. 139–147.
[8]. A. W. Services. (2010) Amazon cloudwatch. Amazon Co.Ltd.[Online].Available: http://aws.amazon.com/cloudwatch/
[9]. K. G. O., A. Francillon, and S. C˘ apkun, "Pay as you browse: microcomputations as micropayments in web-based services," in Proceedings of the 20th intl. conf. on WWW. New York, NY, USA: ACM, 2011, pp. 307–316.
[10]. I. Sharfman, A. Schuster, and D. Keren, "A Geometric Approach to Monitor Threshold Functions over Distributed Data Streams," Proc. ACM SIGMOD Int'l Conf. Management of Data, 2006.
[11]. S. Agrawal, S. Deb, K.V.M. Naidu, and R. Rastogi, "Efficient Detection of Distributed Constraint Violations," Proc. IEEE 23rd Int'l Conf. Data Eng. (ICDE), 2007.