Trust is Good, Control is Better: Creating Secure Clouds by Continuous Auditing

Sebastian Lins, Stephan Schneider, and Ali Sunyaev

Abstract—Cloud service certifications (CSC) attempt to assure a high level of security and compliance. However, considering that cloud services are part of an ever-changing environment, multi-year validity periods may put in doubt reliability of such certifications. We argue that continuous auditing (CA) of selected certification criteria is required to assure continuously reliable and secure cloud services, and thereby increase trustworthiness of certifications. CA of cloud services is still in its infancy, thus, we conducted a thorough literature review, interviews, and workshops with practitioners to conceptualize an architecture for continuous cloud service auditing. Our study shows that various criteria should be continuously audited. Yet, we reveal that most of existing methodologies are not applicable for third party auditing purposes. Therefore, we propose a conceptual CA architecture, and highlight important components and processes that have to be implemented. Finally, we discuss benefits and challenges that have to be tackled to diffuse the concept of continuous cloud service auditing. We contribute to knowledge and practice by providing applicable internal and third party auditing methodologies for auditors and providers, linked together in a conceptual architecture. Further on, we provide groundings for future research to implement CA in cloud service contexts.

Index Terms—Certification, cloud computing, continuous auditing, security

1 Introduction

An increasing number of organizations outsource their data, applications and business processes to the cloud, empowering them to achieve financial and technical benefits due to on-demand provisioning and pay-per-use pricing. However, organizations are still hesitant to adopt cloud services because of security, privacy, and reliability concerns regarding provisioned cloud services as well as doubts about trustworthiness of their cloud service provider [1], [2], [3]. Cloud service certifications (CSC) are good means to address these concerns by establishing trust, and increasing transparency of the cloud market [2], [4]. Several CSC have evolved, such as *CSA STAR* or *EuroCloud Star Audit*. These CSC attempt to assure a high level of security, reliability, and legal compliance, for a validity period of one to three years. However, cloud services are part of an ever-changing environment, resulting from fast technology life cycles and inherent cloud computing (CC) characteristics, like on-demand provisioning and entangled supply chains [5], [6]. Hence, such long validity periods may put in doubt reliability of issued certifications. CSC criteria may no longer be met throughout these periods, for instance, due to configuration changes or major security incidents. Thus, continuous auditing (CA) of certification criteria is required to assure transparent, continuously reliable, and secure cloud services and to establish a trustworthy CSC after the initial certification process is accomplished.

• *The authors are with the Department of Information Systems, University of Cologne, Cologne, Germany.*
E-mail: {lins, schneider, sunyaev}@wiso.uni-koeln.de.

Manuscript received 16 Oct. 2015; revised 4 Jan. 2016; accepted 24 Jan. 2016. Date of publication 27 Jan. 2016; date of current version 31 Aug. 2018. Recommended for acceptance by J.H. Abawajy.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TCC.2016.2522411

Extant research has focused on implementing and evaluating CA of information systems since the early nineties. This progression has included the evolution of architecturally different methodologies, for instance, embedded audit modules [7] and independent monitoring control layers [8], which help to monitor and audit information systems. However, past research has mostly examined CA for internal purposes only. In the context of CC, researchers recently proposed the means to enable third party authorities to audit data integrity [9], data location compliance [10], and changes of cloud infrastructure [11] among others. Aside from these special purpose methodologies, research currently lacks a comprehensive architecture, enabling third party auditors to continuously audit a broad variety of CSC criteria.

We address this gap by conceptualizing an architecture for CA of cloud services, comprising main components, methods, and processes while considering the requirements and needs of main stakeholders. Before conceptualizing an architecture, and thus defining how to perform CA, it has to be analyzed where CA is reasonable. Subsequently, we analyze which CSC criteria should be continuously audited to assure ongoing adherence by performing workshops with cloud service auditors first. Then, we evaluate how these criteria can be continuously audited to identify main auditing components and methodologies for our proposed architecture. Therefore, we build on and extend previous work on CA methodologies by Lins et al. [12]. To ensure applicability of considered methodologies in the CC context, we conducted three expert interviews with cloud service auditors. To link methodologies and to discuss potential architectures as well as to consider stakeholder requirements, we conducted three workshops with cloud service providers, consultants, and auditors as well as ten semi-structured interviews with cloud service customers. Finally, based

upon these findings, we conceptualize an architecture to continuously audit cloud services in a practically and economically feasible manner. Summing up, we focus on the following objectives within this study:

i. Which CSC criteria should be continuously audited?
ii. Which CA methodologies exist and are applicable in the context of continuous cloud service auditing?
iii. How can methodologies be linked together to form an architecture which enables CA?

Our findings reveal that various CSC criteria (e.g., performing regular vulnerability testing and ensuring data integrity) should be continuously audited to assure ongoing certification adherence and to prove secure and reliable services. Moreover, interviews revealed that most of existing methodologies are not applicable for (external) third party auditing purposes. Therefore, providers have to establish an internal auditing department, which manages provision of audit-relevant data. Finally, our conceptual architecture highlights important components (i.e., various interfaces and auditing management modules) as well as processes that have to be implemented. We thereby contribute to practice and research in several ways:

1. We support cloud auditors to classify whether or not a high frequency auditing of their CSC criteria is required. Further on, we illustrate methodologies which can be used by auditors to perform (external) auditing of cloud services as well as by cloud service providers to set up an internal auditing department.
2. We transfer the concept of CA in a new context, provide means and foundations for further research, and demonstrated benefits, challenges, and limitations of CA of cloud services.
3. By providing a first conceptual architecture, we want to encourage auditors and cloud service providers to implement CA techniques, consequently creating trustworthy certifications and services.

This paper proceeds as follows. We first provide a background on CC, CA, and related work, followed by a presentation of our research approach. In Section 4, we briefly evaluate what criteria should be continuously audited. In Section 5, we discuss identified methodologies. In Section 6, we present our architecture for CA of cloud services. We then discuss challenges and benefits in Section 7 and 8 as well as conclude with directions for future research.

2 Background

2.1 Cloud Computing and Certification

Cloud computing enables ubiquitous, on-demand network access to a shared pool of configurable computing resources that can be rapidly provisioned and released with minimal management effort or service provider interaction [13]. These resources refer, for instance, to hardware, development platforms, and applications. CC entails five essential characteristics, that are: provision of (i) on-demand self-service access to (ii) virtualized, shared, and managed IT resources that are (iii) scalable on-demand, (iv) available over a network, and (v) priced on a pay-per-use basis. These characteristics challenge current assessment processes [6]. Therefrom, CC faces a broad range of security issues, including accessibility vulnerabilities, privacy, and control issues as well as issues related to data integrity and data confidentiality [1].

Extant research already proposes certifications and audits as detective controls and good means to assess quality and performance of IT services in procurement processes [2], [4], [14]. A certification is defined as a third party attestation of products, processes, systems, or persons that verifies conformity to specified criteria [15]. Several CSC (e.g., *CSA STAR*) and cloud certification schemes in particular (e.g., *ISO 27017*) have emerged to assure a high level of security, reliability, and legal compliance of cloud services. Recent research suggests that CA is required to deal with the ever-changing environment of cloud services and to increase trustworthiness of CSC [6], [16], [17].

2.2 Continuous Auditing

Continuous auditing is defined as a methodology that enables independent auditors to provide written assurance on a subject matter, using a series of auditors' reports issued virtually simultaneously with, or a short period of time after, the occurrence of events underlying the subject matter [18]. Thus, CA enables auditors to immediately react to changes or events concerning the subject matter and to adjust their auditing reports based on assessment of these changes and events.

The early works of Groomer and Murthy (1989) concerning implementation of embedded audit modules [7] and Vasarhelyi and Halper (1991) regarding usage of monitoring and control layers [8] spawned a research stream of CA. Therefrom, extant literature investigates implementation, transferability, and diffusion of CA in varying domains [19], [20], [21]. Recently, researchers discussed CA of enterprise resource planning systems [22], [23], accounting systems [24], [25], and web services [26], [27]. In the context of CC, research started to propose different approaches to enable third party auditing, for example, methodologies to enable auditors to simultaneously verify integrity of multiple users' data [9] and to assure data location compliance by analyzing audit logs [10]. However, a comprehensive CA architecture, which is able to audit a broad range of CSC criteria and combines various methodologies, is still missing.

2.3 Related Work

Extant research has already focused on automation of assessment and certification approaches in the context of service oriented architectures. For example, Lamparter et al. demonstrate how to automatically evaluate whether a web service execution meets contract requirements [28]. Ardagna et al. propose a machine-readable certification, which is issued to the service after validating its reliability properties [29]. Stephanow and Fallenbeck demonstrate how metrics can serve to support continuous validation of generic CSC criteria [17], [30], [31]. Lins et al. reviewed various automated auditing and monitoring methodologies, and briefly evaluate their applicability in the context of cloud computing [12]. Further on, a variety of research focuses on developing and analyzing cloud architectures (e.g., [32], [33]) to enable continuous monitoring of cloud services. Continuous monitoring is defined as ongoing observance and analysis of operational states of systems and applications to provide decision support, detect and diagnose problems, and to

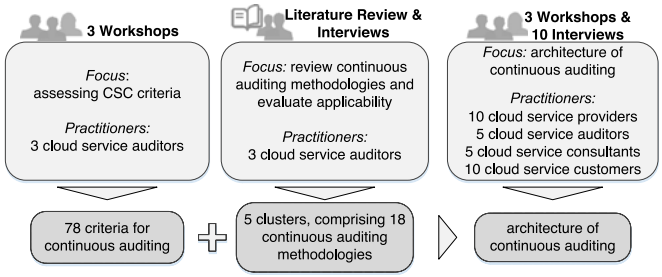


Fig. 1. Overview of research approach.

provide information for further analyses [33] and is performed by service providers or a third party. Consequently, continuous monitoring approaches are designed for internal monitoring purposes only, and gathered monitoring information is kept in-house to be solely inspected by system administrators. Hence, in contrast to CA, monitoring of CC infrastructures does not provide any proof to customers that provided services are reliable and secure.

3 Research Approach

To gain practical insights into CSC auditing processes and to get access to expertise and experience of practitioners, we cooperated with *CloudAuditor*, an auditing company offering global, independent quality and safety inspection services as well as a CSC, which is becoming increasingly relevant across Europe.

To answer the research questions, we apply a three-step research approach (see Fig. 1). We first conducted workshops with practitioners from *CloudAuditor* to assess CSC criteria. Second, we build on and extend previous research on CA methodologies [12] and conducted interviews with *CloudAuditor* practitioners to evaluate their applicability in the context of CC. Finally, we held three workshops with cloud service industry experts as well as ten interviews with cloud customers to elicit requirements and conceptualize an architecture of CA of cloud services.

3.1 Assessing Certification Criteria

Before conceptualizing an architecture for CA, we specify which cloud service parameters, components, and processes have to be continuously audited to assure ongoing secure cloud services. Therefore, we assessed which criteria require a high frequency auditing after the initial certification process was accomplished.

To take a variety of CSC criteria into consideration, we included two different CSC criteria catalogues: the criteria catalogue from *CloudAuditor* (comprising 273 criteria) and the CSC criteria taxonomy from Schneider et al. (comprising 328 criteria) [16]. For further analysis, the authors and practitioners from *CloudAuditor* jointly merged both catalogues to reduce redundancies and irrelevant criteria. As a result, the final criteria collection comprised 326 CSC criteria.

Three workshops were held with *CloudAuditor* practitioners to jointly assess each criterion whether or not a high frequency auditing is required. The workshops lasted about two hours on average. In total, 78 out of 326 certification criteria were marked as a candidate for CA. Finally, for each criterion an auditing frequency was proposed based upon their experience from conducting cloud service audits and their technical knowledge. Results of the joint assessments are presented in Section 4.

3.2 Assessing Continuous Auditing Methodologies

In previous work, Lins et al. performed a comprehensive literature review to identify (semi-) automated auditing methods that are applicable in the context of cloud computing [12]. Their study yields a set of (semi-) automated methods for continuous monitoring and auditing in six clusters. We build on their findings and further extend their literature review by extending their search string and performing a backward and forward analysis. Therefrom, 18 CA methodologies from 66 research articles were extracted, which will be discussed in Section 5. A detailed explanation on why and how we extend their findings can be found in Appendix A, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/ TCC.2016.2522411.

After identifying methodologies, their practical applicability in context of cloud service auditing needed to be assessed. Therefore, semi-structured one-on-one interviews with practitioners from *CloudAuditor* were conducted. Interviews allow gathering of rich data from people in different roles [34]. Furthermore semi-structured interviews involve use of pre-formulated questions, but allow improvisation for emerging topics during conversation as well. In total, three semi-structured interviews were conducted with two security analysts and one certification consultant, lasting about 80 minutes in average and 4 hours in total. Interview guidelines were prepared individually beforehand and started with general questions concerning execution of auditing processes followed by descriptions of selected methods and questions regarding their feasibility and applicability (see Appendix E, available in the online supplementary material). All interviews were approved to be recorded and transcribed afterwards. Applicability assessments were analyzed and are presented in Section 5.

3.3 Conceptualizing an Architecture for Continuous Auditing

To link identified methodologies, to discuss potential architectural concepts and to consider stakeholder requirements, we conducted three workshops with cloud service experts following the qualitative research method of focus group interviews. Focus group interviews enable us to get collective views on a certain defined topic of interest from a group of people who are known to have certain experiences [34]. Furthermore, focus groups allow participants to engage in thoughtful discussions, hence generating practical oriented and rich data. During these workshops, the concept of CA was lively discussed and exemplarily transferred to individual use cases of practitioners. In total, ten cloud service providers, nine cloud service auditors, and five cloud service consultants participated. The different cloud service providers operate on a national and global scale, providing infrastructure, platform, and software as a service. Providers' sizes ranged from medium to large enterprises. Auditors have multi-year experience in conducting cloud service, infrastructure as well as data security and privacy audits. Further on, auditors are employed by large auditing or certification authorities, or work as independent auditors. Finally, participating consultants advise cloud customers when

choosing cloud services as well as providers when deciding whether to get certified or not. A workshop lasted on average 4 hours and 30 minutes and all three workshops lasted 15 hours in total. Since no cloud service customer participated during these workshops, consultants and providers were asked to represent a customer's perspective and to report on their experience with customers. In addition, we conducted 10 semi-structured one-on-one interviews with cloud service customers. Interviewees are IT managers from medium to large enterprises and different sectors including IT, health, trade, and finance, which use various cloud service models. An interview lasted on average 60 minutes. Appendix E, available in the online supplementary material, illustrates interview guidelines for (focus group) interviews with auditors, providers, and customers.

Interviews were recorded, transcribed, and analyzed by three researchers independently, applying qualitative data coding techniques [34] (software used: *ATLAS.ti 7*). We followed a two stage coding approach: first performing open coding and second axial coding. Our initial stage of analysis (open coding) aimed at identifying new components, processes, and relationships based on data collected. With the goal to conceptualize an architecture, we were particularly interested in understanding potential components and processes as well as relationships and requirements which have to be considered when conceptualizing an architecture. On the second stage of analysis (axial coding), we used components, methods, and architectures assessed in the Section 5 while analyzing transcripts to confirm that these accurately represent interview responses and to explore how they are related to findings from the open coding stage. Finally, based on insights gained during this coding analyses and practitioner discussions during workshops and interviews, a conceptual architecture of CA were derived comprising of necessary processes and components to assure ongoing certification adherence.

4 Continuous Auditing Criteria

Existing CSC represent only a retrospective look at the fulfillment of technical and organizational measures at the time of their issuing. CSC criteria may no longer be met throughout certification validity periods. Current CSC are facing several drawbacks when assuring ongoing certification adherence, including:

a. *Inherent cloud computing characteristics.* Cloud services are part of an ever-changing environment, resulting from inherent CC characteristics, like on-demand provisioning and entangled supply chains. Furthermore, cloud services are characterized by fast technology life cycles compared to other industries.
b. *Ongoing architectural changes.* Hardware or software configuration changes as well as changing sub service providers might lead to certification violations or security vulnerabilities. Such security vulnerabilities could go undetected for a long time without appropriate monitoring mechanisms. Especially in case of performing agile software development and providing cloud applications, current certifications are lacking capabilities to observe and deal with continuous deployments.

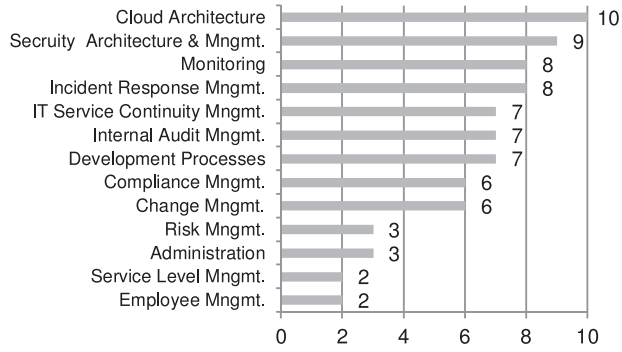


Fig. 2. Categories of criteria that were marked for CA.

c. *Environmental threats.* Changes in the CC and IT environment, for example emergence of new vulnerabilities, require providers to adapt their services to cope with emerging challenges. Major security incidents may threaten the service or reveal harmful vulnerabilities, which in turn void a certification.
d. *Changes in legal and regulatory landscape.* The legal and regulatory landscape of cloud services is highly dynamic since existing laws are currently adjusted, and new laws are being proposed to cope with challenges resulting from the digital transformation of society and continuous changes in IT. Just recently, the Safe Harbor data sharing agreement between the European Union and the United States was questioned. These dynamics might change responsibilities of both cloud service customers and providers as well as require certification criteria to be successively updated.
e. *Deliberate discontinuance.* A cloud service provider might deliberately discontinue adherence to CSC criteria to achieve benefits (e.g., cost savings).

Assessments during the workshop revealed that 78 of 326 certification criteria should be continuously audited since they are affected by at least one of these drawbacks. For example, a CSC criterion states that '*source code reviews should be performed regularly to identify possible vulnerabilities and security issues when developing software*'. Since this criterion implies actions, which have to be performed on a regular basis, cloud service providers might deliberate discontinue fulfilling this criterion to save costs. Appendix B, available in the online supplementary material, provides a checklist to assess CSC criteria and corresponding examples. Additionally, Appendix C, available in the online supplementary material, summarizes findings regarding potential frequencies of CA.

Criteria were further grouped into different categories based upon their requirement contexts. Fig. 2 lists these categories and the number of criteria contained. Interviews with customers confirm that especially criteria ensuring service availability, data integrity and location, a secure access management, and data encryption should be continuously audited. These criteria are reflected by the following categories, which will be briefly outlined.

First, criteria of category *Cloud Architecture* ensures ongoing network security, performing backups and assure secure multi-tenancy capabilities. Second, criteria in category *Security Architecture and Management* necessitate performing

TABLE 1
Clusters of Identified CA Methodologies

Computer-Assisted Auditing Tools and Technologies (CAATT)
Packaged and automated auditing processes that enable auditors to extract, sample, and analyze auditees' data.
Evidence Gathering Mechanisms
Mechanisms to gather and store electronic evidence and information, for example, embedded audit modules or digital agents.
Auditing System Architectures
Architectural concepts to design and to perform CA.
Log Inspection
Inspect logs, which contain information about system operation.
Data Integrity Validation
Methods to audit integrity of customer data stored in a cloud.

continuously vulnerability analysis and assuring encrypted data storage, data confidentiality and integrity. Moreover, *Monitoring* compromises criteria, which require ongoing monitoring of service availability, cloud components, and networks. *Incident Response Management* contains criteria that require providers to receive and process incident messages in a timely manner. Further criteria belonging to category *IT Service Continuity Management* require providers to test, extend, and update service and business continuity plans regularly. Criteria assigned to category *Internal Audit Management* necessitate providers to audit potential subproviders, perform and evaluate technical audits. Concerning *Development Processes*, documenting code, performing code reviews, and assuring secure development processes should be continuously audited. Ongoing *Compliance Management* assures compliant data location and service adjustments due to changes of legal or regulatory requirements. *Change management* implies performing (security) tests before integrating new hardware components and software as well as performing patch management processes. Furthermore, *Risk Management* requires providers to perform ongoing risks analyses, reviews, and updates of risk management plans. Criteria contained in category *Administration* ensure performing regular administration tasks, for example, deletion of inactive user accounts. *Service Level Management* implies monitoring and reporting of service level agreements adherence. Finally, *Employee Management* contains criteria that recommend performing regular employee trainings.

This brief outline approves that various CSC criteria should be continuously audited to assure permanent secure cloud services since they can be affected by different influences (e.g., environmental threats). These criteria address different areas, demand individual audit-evidence, and hence require various CA methods.

5 Continuous Auditing Methodologies

In this section, identified CA methodologies are presented and discussed according to their applicability in CC contexts. Methodologies were clustered regarding their objectives and application contexts (see Table 1). Appendix D, available in the online supplementary material, provides an overview of identified methodologies.

5.1 Computer-Assisted Auditing Tools and Technologies

Since the 1980s, researchers and practitioners have developed various computer-assisted auditing tools and technologies which might support CA [35], [36]. Computer-assisted auditing tools can be used by an auditor as part of their audit procedures to connect to an auditee's information system, automatically extract, sample, and analyze necessary data [37]. These tools comprise generalized auditing software, electronic working papers, and tools for fraud detection among others [38], [39]. More importantly, CA functions were recently added to these tools. However, existing computer-assisted auditing tools are mainly used in and developed for accounting contexts [35], [36], [39]. Hence, their applicability in CC contexts might be questionable. Likewise, interviews revealed that in CSC auditing practice computer-assisted auditing tools are mainly used to support technical security analyses, for instance, penetration tests and vulnerability scans.

Regularly performing penetration tests and vulnerability scans are recommended by auditors and demanded by customers to validate adequate security mechanisms and to identify system vulnerabilities. By attempting to execute prohibited behavior or attacks on vulnerabilities, auditors can verify that such behavior is prevented or detected and compensated. A broad variety of corporate and open-source tools exist to support efficient penetration testing and vulnerability scanning (e.g., *Qualys*). These tools provide a variety of (semi-) automated functions (e.g., scan network ports, identify running services, analyze and test well-known vulnerabilities), and can be individually configured based upon an auditee's context. Performing extensive penetration tests on a continuous basis (e.g., weekly) might be limited since they still require high manual efforts. Thus, future research should evaluate how to design automated auditing processes that use penetration tests to validate security measures.

Aside from emergence of a broad variety of tools, important technological advances enhanced technological feasibility of CA [40]. The introduction of XML, the development of the *'Extensible Business Reporting Language'*, and its extension *'Global Ledger'* enabled a platform independent, efficient, and effective exchange of business information over the Internet [27], [40]. Practitioners recommend using XML-coded data when exchanging data between auditee and auditor, because XML-coded data is standardized, well-structured, and can be processed quickly. Similar, a broad variety of formal languages can be used to facilitate CA. For example, the *'Open Vulnerability and Assessment Language'* can be used to enable semi-automated patch and configuration management validation, and vulnerability assessments [41].

5.2 Evidence Gathering Mechanisms

Several automated methodologies have been identified to enable auditors to gather electronic audit evidence. The most frequently mentioned component to gather audit evidence is an embedded audit module (EAM). EAM are special purpose functions, programs, or other code objects that are embedded into auditees' information systems and supervise all of audit-related data in real-time [7], [35], [42], [43], [44], [45], [46], [47]. One of the most important advantages of

EAM is that they automatically act as triggers and inform the auditor when suspicious events appear, thus eliminating the need for high frequency assurance queries [7], [35], [42]. Recently, organizations have begun to equip EAM with artificial intelligence to expand their capabilities [46]. However, EAM are more vulnerable to manipulation, especially by auditees' employees who have necessary access privileges to interfere [7], [42]. In the context of CC, an EAM might be used, for example, to monitor reliability and availability [29]. Nonetheless, usage of EAM for continuous cloud auditing may be limited because incorporation of EAM into a cloud architecture (that is distributed across different datacenters and locations) requires a complicated, expensive development and customization process [7], [12], [24], [42]. More importantly, practitioners assess that usage of EAM might not be feasible. First of all, most auditees are not willing to permit auditors to integrate third party IT components due to security and privacy concerns since they might cause new security vulnerabilities or disturb system operation. Auditees might even fear data theft or corporate espionage. Moreover, integrating EAM might violate internal compliance requirements or corporate security policies. Practitioners empathized that usage of EAM seems to be exclusively feasible, if these EAM require minimal access privileges and analyze non-confidential data. Hence, practitioners and researchers need to evaluate how EAM might be implemented while ensuring confidential data gathering and exchange with auditors.

Alternatively, digital agents can support auditing processes [11], [21], [35], [48], [49], [50]. Digital agents (also referred to software, autonomous, or intelligent agents) are software objects that achieve individual goals by autonomously performing actions and reacting to events in a dynamic environment. They are characterized by having different degrees of artificial intelligence and mobility (the ability to travel from one platform to another) [35], [49]. In contrast to EAM, digital agents remain on the auditor's side and are only deployed to the auditee's infrastructure during auditing processes. Digital agents are supposed to automatically perform activities that are traditionally undertaken by human auditors, for example, collecting audit evidence, and validating certification criteria [21], [35], [49]. Typically, audit tasks are performed by a team of agents, which are hierarchically structured [11], [51]. Through their artificial intelligence, mobility, and individual, autonomous acting, they seem to be very suitable for continuous cloud service auditing, especially when comparing digital agents to EAM. However, high efforts and expenses for agent development and implementation as well as possible negative impacts on system performance have to be considered [35]. Similar to EAM, usage of digital agents has been evaluated critically by practitioners: "*I think that customer acceptance to permit digital agents is very low because with these agents you implement untrustworthy software into your cloud systems*" [Auditor]. Especially agent deployment interfaces might be a highly valuable target for attackers to compromise auditees' and auditors' systems. Thus, future research should evaluate how to address potential security vulnerabilities when using (third party) digital agents (e.g., sharing encryption keys between agents and data sources to enable authentication [52]).

Furthermore, an interceptor can be applied as a wrapper that is used to encapsulate information systems or IT components [24], [53]. They can monitor data flowing into and out of systems, therefore enabling CA. Interceptors can be configured to validate accordance with implemented business logics and certification criteria [54]. Contrary to EAM, interceptors usually operate independently of information systems. Hence, they can be implemented in any phase of a software life cycle, and detailed knowledge about auditees' information systems are not necessary to initiate an interceptor. Currently, different vendors offer a variety of tools to implement interceptors on different system layers to capture messages (e.g., *Apache Axis handler* for the middleware layer) [24]. Practitioners are currently using interceptor tools (e.g., *Burp Suite*) to intercept data streams between cloud servers and their web browsers to identify and test security vulnerabilities. Auditors suggest that interceptors can be used for CA of cloud services. However, when implementing interceptors for CA, one has to filter and adjust the amount of data that is actually analyzed to prevent performance losses, security and privacy concerns (e.g., fear of monitoring employees).

Further on, CA systems may incorporate different data, text, and process mining techniques, which are performed on a regularly basis to extract audit evidence [46], [55], [56]. Data mining techniques try to discover patterns in large sets of data and to detect irregularities. Text mining involves discerning patterns from text to detect deception and fraud, and can be applied, for example, to email, discussion groups, media, and in general to the Internet. These mining techniques can be used, for example, to detect changes in auditees' system architecture. Moreover, auditors need to identify and evaluate external changes, for example, emergence of security threats or vulnerabilities, which might trigger re-auditing events or alerts. Mining techniques can be used to assess open vulnerability databases (e.g., *Common Vulnerability and Exposures Database*) to expose unknown vulnerabilities and system weakness configurations that may cause system crashes and malfunctions [55]. Finally, by using process mining techniques auditors can gain insights into how processes are being undertaken by analyzing a vast amount of data that is routinely gathered and stored in event logs [56]. Process mining compares actually logged process with a designed process model. Interviews revealed that such process mining techniques might bear great potential to continuously audit and confirm process executions. Yet, comparison of derived and designed process models might be limited since most auditees do not provide models in suitable, machine-readable formats to enable an automated model comparison. Hence, future research should evaluate how process mining techniques might be used in the CA contexts, for example, by using training log files to automatically and continuously create process models [57].

5.3 Auditing System Architectures

To enable a continuous evidence extraction and transmission, a communication model is required. An auditor's system can be efficiently connected to auditees' systems, for example, by using the *Simple Object Access Protocol* to exchange messages, or by using the *Common Object Request*

Broker Architecture as a middleware to gather information from heterogeneous auditees' applications [58]. Audit-relevant data can be transferred at predetermined intervals and then stored in supplementary databases, for instance, in audit data marts [22], [45], [51]. Audit data marts are small, mostly auditee-independent data repositories in which relevant data is automatically stored, enabling real-time data access and automated data analyses [45].

Aside from individual mechanisms to gather audit evidence, researchers have developed several comprehensive CA system architectures. A monitoring and control layer can be implemented as an independent auditing system [23], [42]. This system forms an overlay on top of a set of existing systems and utilizes a middleware layer to provide integration between loosely coupled applications such as auditees' service applications and legacy systems [23], [25], [59]. Data from integrated applications can be extracted and compared to a predefined auditing rule-set, and detected violations might automatically trigger an alert. This system is owned and operated by the auditor, thus data retrieved can be presumed to be tamper-proof [23], [42]. Applicability of monitoring and control layers in CC contexts might be limited due to distributed cloud infrastructures.

Besides, agent-based CA architectures are common in literature and practice. Under this architecture, a digital agent is initiated to represent a certain audit procedure and dispatched to different auditees' systems [35], [51]. A flexible (e.g., platform independent) and adaptable (e.g., agent can be deployed as required) agent-based architecture facilitates gathering audit evidence in distributed and heterogeneous auditees' systems [60]. Hierarchically structured teams of agents will perform planned audit operations, for example, interacting with an auditee's system and retrieving necessary audit evidence, testing effectiveness of business processes, or mining data to analyze and identify fraud behavior [11], [51]. In contrast to usage of monitoring and control layers in cloud contexts, agent-based CA architectures enable a flexible deployment and transmission of agents across different cloud infrastructures and locations.

Furthermore, CA can be implemented as a set of web services that reside within an auditor's computing environment [26], [27], [40]. Each auditing function is therefore represented as a web service which can be invoked to continuously audit an auditee's system [27]. Usage of web services for auditing enables new business models for auditors, for example, cloud customers paying a service fee for invoking an auditor's web service to continuously retrieve assurance reports. In this regard, an incident detection web service was developed to enable customers to observe individual specified security and monitoring policies [11].

When performing CA, a huge volume of data, exceptions and reports may be generated, thus threatening audit efficiency. To counteract this issue, it is recommended to implement decision support systems (DSS) [46]. Such systems might enhance CA by aggregating information from many different sources (e.g., agents or minded data), reacting on auditee reports, and by efficiently and automatically deciding to take actions or to alert the auditor. Consequently, DSS ultimately reduce workload of auditors. Future DSS may even evolve to intelligent and adaptive audit process systems, which automatically adjust audit tests based on gathered data and unexpected events [46]. Interviews revealed that DSS are currently not used during CSC processes. Nonetheless, auditors endorse the concept of using these systems to support and to automate their auditing.

5.4 Log Inspection

Interviews revealed that auditors agree to inspect logs, which are routinely created during monitoring operations by services providers to assess certification adherence. In the following, exemplary log inspection techniques will be presented. To monitor execution of applications, abstract execution logs can be inspected by using heuristics-based log inspection techniques [61]. Such techniques can inspect log lines with limited format requirements and can scale up to process log files, which contain millions of log lines. Supposedly, such a method can be used to automatically and continuously check whether different applications are actually running on a cloud infrastructure, for example, malware protection or anti-virus software [12]. Likewise, unstructured logs can be automatically analyzed, for example, to detect system anomalies [57]. To analyze unstructured logs, data mining techniques can be used, comprising learning and detection process. During learning processes, ideal models that represent the normal executional behavior of the system are derived based upon training log files. In detection processes, new input logs are compared to the ideal models to automatically detect anomalies. Further on, logs can be analyzed to assure protection of customers' privacy a posteriori [62]. To implement such techniques, a policy language for expression of privacy preferences (e.g., access regulations) and an automated process for checking adherence to policies (e.g., tree pruning strategies [63]) have to be implemented [62].

5.5 Data Integrity Validation

As cloud service customers do not longer possess their data locally, assuring that their data is being correctly stored and integrity is maintained in cloud environments is of critical importance. Data integrity may be threatened by, for example, malicious insiders, data loss, technical failures, and by external attackers [64]. To create trustworthy cloud services, auditors should continuously validate that data integrity is maintained.

A wide range of research currently addresses the question on how to assure data integrity by a third party in CC contexts. Especially hashing techniques have been identified as adequate methods for monitoring integrity of large amounts of data [65], [66], [67]. These techniques enable auditors to simultaneously verify integrity of multiple users' data, which is important in multitenant cloud environments with many users operating at the same time. Moreover, simultaneous monitoring of multiple and hybrid clouds, and multiple owners is feasible [65]. Aside from that, some methods support dynamic data operations on a fine-grained level, thus, minor data changes are considered when validating data integrity [67]. Data security and privacy has to be ensured when validating data integrity in cloud environments, for example, by implementing cryptography [65], authentication [64], or authorization techniques [66], [67]. Furthermore, auditors can reduce communication and computation cost by using periodic sampling audits [66], or moving computational operations onto cloud servers [65]. Scenarios in which cloud users

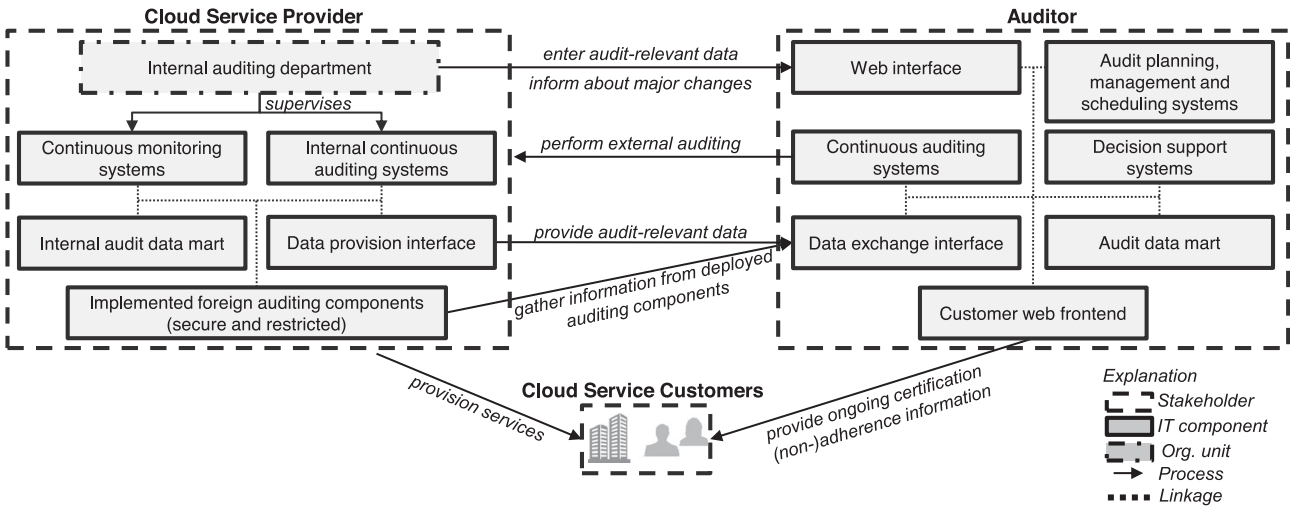


Fig. 3. Conceptual architecture for continuous auditing.

are sharing data as a group require adjusted integrity validation checks since auditors may be able to reveal confidential information of the group by traditional approaches (e.g., which user in the group is modifying data most) [9], [68]. Auditors can use private group keys as additional file signatures and index tables to ensure data integrity in shared data contexts [9], [68].

When stored data is archived, it remains necessary to ensure its integrity for disaster recovery or to assure compliance with legal requirements [69]. To perform automatic integrity checks, a data integrity protection scheme is proposed [69]. Given an archive file, it can be encoded into code chunks, which are distributed over and stored on a number of servers. An auditor can ask for randomly chosen parts of remotely stored data, and run a probability checking protocol to verify data integrity.

Identified methods form a comprehensive sample for enabling continuous, secure, and privacy-preserving auditing of cloud storage data integrity by third parties.

6 Conceptual Architecture

By analyzing extant literature on CA, interviewing auditors, providers and customers, an architecture for CA of cloud services was conceptualized (see Fig. 3) and will be described in the following.

6.1 Data Gathering

CA requires auditors to gather and assess comprehensive data sets on a regular basis. Interviews and workshops revealed that (external) third party access to audit-relevant data, and therefore auditors' data gathering capabilities, are limited due to technical, organizational, and legal reasons. First, technical limitations and barriers might hamper auditors to gather necessary auditing information by themselves. Integration of additional monitoring systems, matching auditees' heterogonous data formats and legacy systems, requires extensive modifications to auditees' systems, which can be quite expensive to implement, especially post hoc. More importantly, most service providers are not necessarily willing or obligated and may be even resisting to integrate auditors' techniques into their systems. Second, efficient data gathering and monitoring requires extensive knowledge about organizational processes, structures, system architectures; nonetheless an auditor's knowledge about an auditee's system and processes is limited due to the nature of focusing on potential security problems and her independence. Likewise, auditors are hesitant to externally interfere with auditees' systems to prevent security vulnerabilities. Third, due to legal requirements, gaining access to required data and systems might be limited for third party auditors as well.

To cope with these limitations, most audit-relevant information has to be gathered by the provider herself, and subsequently be made accessible for auditors—according to practitioners. Hence, performing continuous monitoring by service providers forms a prerequisite for the provision of audit-relevant data, and for auditors to perform efficient CA. Nonetheless, auditors can gather additional data by performing limited external CA.

6.1.1 Continuous Monitoring and Internal Auditing

Providers have already equipped their service centers with sophisticated monitoring technologies to gather service data and quickly detect malicious attacks, failures, and outages. Leveraging collected data for the purpose of CA as well is beneficial. Yet, participating in CA requires providers to use comprehensive *(continuous) monitoring systems* to ensure that all audit-relevant data is up-to-date, accurate, and available. Therefore, cloud providers should implement an extensive cloud logging framework as suggested by Ko et al. [32]. To adhere to CSC criteria presented in Section 4, continuous monitoring operations should at least comprise gathering data by monitoring of physical resources and virtualized environments, security and privacy monitoring as well as service level monitoring.

In addition to performing extensive monitoring processes, a provider might implement *internal (continuous) auditing systems* to gather monitoring data across different systems, and to aggregate and anonymize (monitoring) data, and format data according to auditors' needs. Since auditing methodologies presented in Section 5 are mostly developed for internal auditing contexts, they can be implemented by service providers as well to gather audit-relevant data internally. For instance, by deploying internally a team of digital agents or

implementing a monitoring and control layer, administrators can gather data across different cloud monitoring tools and store findings inside an *internal audit data mart*.

Gathering audit-relevant data by the service provider herself leads to several advantages compared to auditing that solely relies on external auditing. First, auditee resistance will decrease, and acceptance will increase when auditors do not interfere directly with auditees' systems. Second, providers' employees possess or can easily access required knowledge about internal processes and cloud systems. Third, audit-relevant data and information can be gathered and processed internally, hence, reducing security and privacy concerns. Fourth, instead of implementing standardized or inappropriate third party modules and software, an auditee can implement proprietary and customized internal auditing techniques aligned to their individual cloud architecture. However, a provider has to ensure that appropriate monitoring and internal auditing resources are allocated and integrated into daily operational management, and employee responsibilities are settled. Thus, providers have to adjust their organizational structures to meet CA prerequisites and criteria.

6.1.2 External Continuous Auditing

To gather additional audit-relevant data, auditors can perform limited external CA. In general, cloud components that are connected to the Internet can be (continuously) tested and scanned. Therefore, auditors should implement *continuous auditing systems*, comprising various auditing methods to perform (semi-)automated auditing processes for different scenarios. Hence, for example, performing penetration testing, external vulnerability scans, and using interceptor tools to analyze cloud systems, service availability, and encryption. Likewise, presented data integrity checks (see Section 5.5) can be performed externally. Further external assessments can be performed based on criteria requirements. For example, assuring that a *'security incident handling team has to be available 24 h, seven days a week'* by performing automated telephone calls or automatically sending predefined trouble-shooting tickets and assessing responses. Moreover, auditors have to implement *systems to support audit planning, management and scheduling* to coordinate CA processes and to enable fluent and automated execution of auditing functions.

Practitioners empathized that implementation of foreign auditing components is limited but might be applicable in some cases. For example, a provider can implement a foreign component (e.g., EAM or digital agent), which requires solely minimal access privileges and only analyses non-confidential data. Therefrom, auditors might extract additional audit-relevant information from cloud services.

6.2 Data Exchange

Besides gathering data, providers have to manage a provision of audit-relevant information to ensure ongoing data exchange with auditors. Therefore, service providers need to establish an *internal auditing department*, which manages and supervises the gathering, processing, provision and transmission of audit-relevant information. This internal auditing department forms a linkage between provider and auditor when performing CA.

Practitioners suggest different data exchange approaches to assess ongoing certification adherence. First, providers might incorporate defined *data provision interfaces* to enable auditors accessing relevant data. Different types of data provision interfaces might be implemented, for example, a user interface (i.e., a simple web frontend presenting audit-relevant data) or a standardized application programming interface (i.e., XML, JSON, or Perl interfaces). On the other hand, auditors might offer *data exchange interfaces*, for instance, a *web interface* to upload or to enter auditees' data or to inform auditors about major changes. Second, auditees might transmit monitoring logs or exported data from existing monitoring systems (e.g., *Nagios*) to auditors. Finally, auditors might request auditees to provide reports according to defined frequencies. For example, when validating adherence to the exemplary criterion *'a provider should regularly perform reviews of firewall rules'*, then an auditee can upload a short report that comprises various information, for instance, date, firewall policy version, number of offending firewall rules, initiated operations, and changes made.

6.3 Data Analysis and Presentation

Since data is provided on a high frequency, data should be stored by auditors in an *audit data mart* to enable automated data analysis. Therefrom, auditors should implement suitable *decision support systems* to improve audit and analyses efficiency, expedite decision-making processes, and to cope with potential alarm floods. DSS can be used to aggregate gathered information as well as efficiently and automatically decide to take actions or to alert auditors. Furthermore, these DSS might trigger additional auditing operations based upon external changes, for instance, announcement of new viruses or software vulnerabilities (e.g., *Heartbleed vulnerability*). Nonetheless, customers demand that auditors should manually validate auditing results on at least regular basis to ensure that results are not falsified due to technical errors.

Interviewees put high emphasize on customer enlightenment when performing CA to counteract customers' fear of loss of controls and to counteract the impression of using a black box when provisioning cloud services. Therefore, it is important to continuously publish auditing information to prove ongoing certification adherence and to increase transparency about cloud services. A fully transparent CA process is especially demanded by customers to increase trustworthiness. Practitioners recommend providing a user interface (e.g., a *web frontend for customers*) to inform customers about performing CA processes and ongoing certification (non-)adherence. In addition, they recommend informing customers about how and when data was gathered and analyzed to increase comprehensibility and accountability. More importantly, in cases of critical certification violations or major security incidents, customers should be automatically informed by auditors. Customers demand auditors to send periodic auditing reports that comprise a summary of performed auditing processes and suspicious incidents. Further on, customers should be able to configure frontends according to their needs, for example, choosing displayed criteria and type of graphical representation (e.g., chart or graph). Likewise, customers might be provided with functions to request renewed auditing, or to report identified issues or reasons for criteria

non-adherence. Auditors might implement these functions as web services, requiring customers to pay an invocation fee (i.e., see [11]). Finally, auditing results should be archived for certain periods since auditors as well as customers are interested in comparing current with past data. For example, customers might want to look up data of the last month to check for CSC criteria adherence. For data analysis purposes, comparison of current data with historic data can aid the auditing service to learn and configure exceptions and alert patterns (e.g., rule-based configurations based on deviations from historic data).

6.4 Continuous Process Adjustments

The process of CA has to be continuously adjusted to cope with dynamics of an ever-changing and hostile environment. On the one hand, emerging environmental threats or changes in legal and regulatory landscape might induce auditors to adjust their auditing scope, for example, by adding new certification criteria. On the other hand, architectural changes of cloud services (e.g., adding new service functionalities) can cause providers and auditors to adjust their monitoring and auditing processes. Therefore, providers might incorporate the concept of CA into their change management processes to inform auditors on major changes.

7 Challenges

Discussion with practitioners revealed that major challenges have to be tackled before the proposed architecture can be applied in practice.

7.1 Risk of Audit-Data Manipulation

Provision of audit-relevant data by a provider herself has one challenging drawback: the risk of data manipulation. Providers might modify provided data to assure ongoing certification adherence. Preventing cloud service providers to manipulate or euphemize audit-relevant data is an important prerequisite to ensure that CA is trustworthy and reliable. Consequently, providers have to establish secure logging mechanisms which achieve a high degree of log integrity and confidentiality. In order to achieve this, we can build on findings from research area of cloud forensics. Cloud forensics is defined as the application of scientific principles, technological practices to reconstruct past cloud computing events through identification, collection, preservation, examination, interpretation, and reporting of digital evidence [70].

Researchers have proposed various procedures to deal with challenges of cloud forensics (i.e., malicious cloud service providers manipulating log files), ultimately enabling third party investigators to collect and analyze relevant data [71]. Cloud service provider can implement appropriate log adapters to extract and transfer log entries from different logging sources (e.g., hypervisor) to a central logging component [72]. This central logging component transforms log entries into a secure, encrypted and uniform log type. To prevent internal log manipulation, a trusted third party module (e.g., hardware or virtual module [71]) can be implemented that provides secure log encryption functions. Similar, various schemes are proposed (i.e., homomorphic encryption) and evaluated using open-source cloud computing platforms to ensure privacy and confidentiality of log data [73], [74].

Further on, one way of revealing data manipulation is to establish a chain of custody for digital evidence [75], which represents a roadmap that shows how data was collected, analyzed, and preserved in order to be presented as evidence in court. Moreover, several procedures are recommended to gather trusted audit-relevant data, including remote data acquisition over trusted and secure channels, usage of management planes, preforming live forensics on systems in running state, as well as snapshotting a clone of a virtual image among others (cf., [71] for a detailed comparison). Nonetheless, cloud forensics procedures will vary according to service and deployment model of cloud computing [74]. For example, Software- and Platform-as-a-Service models inherit very limited control over process or network monitoring, whereas in Infrastructure-as-a-Service settings some forensic friendly logging mechanism might be deployed. Future research should evaluate how existing procedures from cloud forensics research can be applied to enable CA.

Workshop participants and customers assessed a low likelihood of internal modification since continuous modification constitutes high expenditures. In addition, data manipulation requires a provider to store data volume twice; first unmodified data for internal evaluations, second modified data for auditors and customers. Finally, customers might reveal tampered data when using the service (e.g., tampered availability rate). Yet, customers as well as providers recommend that auditors should randomly perform validation tests on regularly basis to prevent data manipulation or reveal tampered data.

7.2 Security Challenges

Providers and auditors have to face several security challenges when providing and transmitting audit-relevant data, including protection of deployed (data exchange) interfaces, authorization of third parties, security of data transmission, and user interfaces as well as achieving high levels of data integrity and confidentiality. Moreover, auditors form a high valuable target for attackers since they receive data from different and mostly large-sized cloud service providers (due to high costs of CA).

7.2.1 Confidentiality Issues

Assuring confidentiality refers to preserving authorized restrictions on access and disclosure, including means for protecting personal privacy and proprietary information [76]. When data is transferred to auditors or presented to customers, privacy of audit-relevant data has to be ensured to prevent leakage of sensitive or security-relevant information. Therefore, data must be anonymized or filtered respectively. In this sense, providers have to precisely differentiate system monitoring data and cloud customers' data. Revealing sensitive customer data might breach service level agreements and consequently lead to financial compensation. Moreover, exchange of relevant data through using interfaces require providers or auditors to implement robust and secure access control systems and encryption mechanisms for data transmission. Attackers might perform brute force or man-in-the-middle attacks to retrieve sensitive data. Finally, providing sensitive cloud service data bears the risk of malicious auditors, who

might abuse audit-relevant data. Therefore, auditors have to prove that data is kept confidential.

7.2.2 Integrity Issues

Providing integrity refers to guarding information against improper modification or destruction and includes ensuring information nonrepudiation and authenticity [76]. In the context of CA, ensuring integrity and guarding information against improper modification by external as well as internal subjects have to be considered. Attackers might be interested in targeting interfaces and frontends to modify provisioned and presented data. A modification of data might affect an auditor's assessment of criteria adherence, and thus might result in certification non-adherence or customer dissatisfaction. Likewise, attackers might tamper data, which is presented to customers to indicate bad service behavior. Ultimately, these attack scenarios can lead to loss of reputation or cancelation of contracts. Subsequently, providers and auditors need to achieve a high integrity of information and establish security mechanisms.

7.2.3 Availability Issues

Ensuring availability refers to ensuring timely and reliable access to and use of information [76]. In the context of CA, availability of cloud systems and provided interfaces has to be ensured. First, performing continuous monitoring and auditing process (e.g., ongoing data gathering, analysis and aggregation operations) might have a substantial performance impact on cloud services. Likewise, failures in these operations might lead to disturbance of cloud service operation. Hence, CA might threaten cloud service availability. Second, when audit-relevant data is provided via defined interfaces, providers have to assure availability of them. Attackers might target interfaces, for example, by performing distributed denial of service attacks to disturb the process of CA. In worst cases, this might lead to non-adherence of CSC criteria, since auditors are lacking corresponding audit information. Finally, providers have to assure that provided user interfaces for customers are available.

7.3 Automation & Cloud Service Individualism

Performing CA requires a high degree of automation. Yet, current CSC are mostly based upon manual auditing operations, for example, performing interviews and manual security tests as well as analyzing service and architecture documentations. Automation of these operations require a strong formalization, which is currently not achievable for every process [42]. Human auditors still need to manually validate specific criteria because some weaknesses might remain unrecognized on automated validation systems. Further on, research suggests that such an automation of processes is likely to be incremental rather than disruptive since auditors will likely attempt to first automate existing processes rather than developing technology enabled auditing processes [77].

In addition, CA requires auditors to automatically assess comprehensive audit-relevant data. However, auditors are faced with a high individualism and complexity of an auditee's cloud service systems, resulting from customized or legacy systems as well as incorporated third party services. Thus, for example, data logs are in heterogeneous formats and hence, it is difficult to automatically examine and analyze log evidence. Therefrom, auditors need to adjust their auditing and analysis methodologies based on context and capabilities of an auditee, which in turn hampers automation of auditing processes.

To deal with this individualism, auditors should develop a comprehensive metric and key performance indicator collection (see for example [17]), which can be used to evaluate criteria adherence based on different data inputs. Metrics might be derived and classified according the goal question metric method in a systematic top-down fashion by defining the goal to analyze cloud computing system designs and questions that help achieving corresponding goals [78]. Subsequently, auditors need to implement flexible and standardized auditing systems, which allow them to easily integrate or exclude providers since they might concurrently audit a broad variety of different cloud service providers. Notwithstanding these adjustments to cope with individualisms, auditors have to ensure that comparability between certification results is guaranteed.

8 Benefits

Providers as well as auditors must be motivated to participate in CA, and hence, to enable diffusion of continuous cloud service auditing. To motivate them, perceived benefits must be higher than perceived expenditures. Notwithstanding preceding challenges, CA will bear great benefits for auditors, providers and cloud customers [79].

Auditors can improve their audit efficiency by reducing auditing time and errors due to automated auditing process. Likewise, CA is more cost-effective by enabling auditors to test larger samples and examine data faster and more efficiently compared to their manual predecessors. CA allows auditors to actively detect and investigate exceptions as they occur rather than to react after exceptions have long occurred. Hence, CA can be considered as proactive and enables corrective action to be taken as soon as a problem is detected. More importantly, through timely detection and continuous assurance of certification adherence, CA can improve trustworthiness of auditors' CSC. Finally, auditors can counteract lack of cloud customers' control in CC environments by increasing transparency regarding operations of service providers.

Cloud service providers can benefit by participating in CA as well. First, internal processes and systems can be improved by implementing suitable monitoring and internal auditing techniques, and evaluating continuous feedback about how they are performing. In addition, providers receive ongoing expert assessments about their systems. Therefrom, CA positively effects service and risk management of providers, which was also emphasized by practitioners. Second, improvements and enhancements of cloud infrastructure, software, or processes (e.g., due to agile development)—after the initial certification—can be considered earlier and reflected in the certification report due to ongoing assessment. Finally, providers can differentiate themselves in the cloud market by making their cloud services more transparent to customers. Thus, they may gain competitive advantages.

Cloud service customers can benefit when CA is performed. Typically, cloud environments are characterized by

a lack of control since cloud customers cedes governance to cloud service providers. Especially when storing data in the cloud, customers fear that data could be compromised or leaked since they are lacking transparency about how and where data is stored and processed. CA can counteract this lack of control by increasing transparency regarding operations of providers. Through an increased transparency, CA ultimately tries to increase trustworthiness of customers in cloud services.

9 Conclusion

The ever-changing cloud environment, fast update cycles, and the increasing adoption of business-critical applications from cloud service providers demand for highly reliable cloud services. Continuously auditing such cloud services can assure a high level of security and reliability to (potential) cloud service adopters. However, methodologies to efficiently and continuously audit cloud services are still in their infancy. With our study, a first step to increase trustworthiness of CSC is provided by conceptualizing an architecture to continuously audit cloud services.

9.1 Contribution to Knowledge and Practice

Our findings reveal that various CSC criteria should be continuously audited to assure ongoing certification adherence and to prove secure and reliable services. Interviews revealed that most of existing methodologies are not applicable for third party service auditing purposes. Therefore, providers have to establish an internal auditing department, which provides audit-relevant data to auditors via defined and secure interfaces. Our conceptual architecture highlights important components (i.e., data provision and exchange interfaces, audit management modules, and customer frontends) as well as processes (e.g., sending reports, inform about major cloud changes, adjusting processes) that have to be implemented. These findings are relevant for practitioners and researchers.

We support auditors by providing a checklist (see Appendix B, available in the online supplementary material), which enables them to classify whether or not a high frequency auditing of CSC criteria is required, after the initial certification process is accomplished. Therefrom, auditors might start to verify criteria adherence on a higher frequency compared to current practices. Further on, we illustrate methodologies, which might be used by auditors to perform external auditing of cloud services (e.g., validating data integrity). Likewise, cloud service providers might implement presented auditing methods to set up an internal auditing department. By providing a first conceptual architecture, comprising important components to enable CA, we want to encourage auditors to implement CA techniques to create trustworthy certifications as well as practitioners to develop business models, for instance, auditing as a service in these contexts.

We also transferred the concept of CA in a new context, and discussed challenges and benefits of CA of cloud services. By identifying and assessing applicability of existing CA methodologies and conceptualizing an architecture, we identify gaps and means to perform CA of cloud services, hence forming a basis for future research. We want to encourage further researchers to address these issues, and thereby, ultimately create continuously secure and reliable cloud services.

9.2 Limitations

Nevertheless, this study has some limitations. First, even if we derived our architecture based on interviews with various providers, auditors and customers from different organizations, our evaluation regarding applicability of identified CA methodologies might be slightly biased since we evaluated applicability within three interviews with practitioners from one cloud service auditor only. Second, within our conceptual architecture, we do not provide any technical implementation. Instead, we focus on giving a broad outline and insights into current state and issues of CA to motivate researchers and practitioners to engage in these topics. We believe that CA of cloud services is one possible way to address current gaps and issues in CC. It is a step forward to a more trustworthy and transparent CC computing environment.

9.3 Future Research

As the discussion of challenges reveals, there is still plenty of research to do. Further research should focus on developing auditing methodologies adjusted to the CC context, especially concerning validation of security measures and adherence to critical cloud service characteristics (e.g., availability and scalability of services). Likewise, future research should examine how unique cloud computing characteristics influence (continuous) auditing practices. Identified methodologies need to be implemented to prove their practical and economic applicability in cloud environments. Therefore, identified and future methodologies need to be linked to CSC criteria and corresponding metrics to measure criteria adherence. Furthermore, research should focus on evaluations regarding acceptance and benefits of cloud providers when participating in CA as well as drivers and inhibitors for cloud service customers' demand for CA. Besides, future research should clarify how to manage certification violations, and if and how to inform cloud customers about certification (non-)adherence.

Acknowledgments

This research is funded by the German Federal Ministry for Education and Research (grant no. 16KIS0079).

References

[1] S. Subashini and V. Kavitha, "A survey on security issues in service delivery models of cloud computing," *J. Netw. Comput. Appl.*, vol. 34, no. 1, pp. 1–11, 2011.
[2] K. M. Khan and Q. Malluhi, "Trust in Cloud Services: Providing More Controls to Clients," *Computer*, vol. 46, no. 7, pp. 94–96, 2013.
[3] S. Schneider and A. Sunyaev, "Determinant factors of cloud-sourcing decisions," *J. Inform. Techn.*, 2014.
[4] A. Sunyaev and S. Schneider, "Cloud services certification," *Commun. ACM*, vol. 56, no. 2, pp. 33–36, 2013.
[5] S. Cimato, E. Damiani, R. Menicocci, and F. Zavatarelli, "Towards the certification of cloud services," in *Proc. 9th World Congress Serv.*, Santa Clara, CA, USA, 2013, pp. 100–105.
[6] I. Windhorst and A. Sunyaev, "Dynamic certification of cloud services," in *Proc. 8th Int. Conf. Availability, Reliability Security*, Regensburg, Germany, 2013, pp. 412–417.
[7] S. M. Groomer and U. S. Murthy, "Continuous auditing of database applications," *Inf. Syst. J.*, vol. 3, no. 2, 1989.

[8] M. A. Vasarhelyi and F. B. Halper, "The continuous audit of online systems," *Auditing*, vol. 10, no. 1, pp. 110–125, 1991.

[9] B. Wang, B. Li, and H. Li, "Oruta," *IEEE Trans. Cloud Comput.*, vol. 2, no. 1, pp. 43–56, 2014.

[10] P. Massonet, S. Naqvi, C. Ponsard, J. Latanicki, B. Rochwerger, and M. Villari, "A monitoring and audit logging architecture for data location compliance in federated cloud infrastructures," in *Proc. IEEE Int. Symp. Parallel Distrib. Process. Workshops Phd Forum*, Shanghai,China, 2011, pp. 1510–1517.

[11] F. Doelitzscher, C. Reich, M. Knahl, A. Passfall, and N. Clarke, "An agent based business aware incident detection system for cloud environments," *J. Cloud Comput.*, vol. 1, no. 1, p. 9, 2012.

[12] S. Lins, S. Thiebes, S. Schneider, and A. Sunyaev, "What is really going on at your cloud service provider?," in *Proc. 48th Hawaii Int. Conf. Syst. Sci.*, 2015, pp. 1–10.

[13] P. Mell and T. Grance, "The NIST definition of cloud computing," 2011.

[14] S. Pearson, "Toward Accountability in the Cloud," *IEEE Internet Comput.*, vol. 15, no. 4, pp. 64–69, 2011.

[15] International Organization for Standardization, Conformity Assessment – Vocabulary and general principles, 17000:2004.

[16] S. Schneider, J. Lansing, F. Gao, and A. Sunyaev, "A taxonomic perspective on certification schemes," in *Proc. 47th Hawaii Int. Conf. Syst. Sci.*, Big Island, Hawaii, USA, 2014, pp. 1–10.

[17] P. Stephanow and N. Fallenbeck, "Towards continuous certification of Infrastructure-as-a-service using low-level metrics," in *Proc. 12th IEEE Int. Conf. Adv. Trusted Comput.*, Beijing, China, 2015, pp. 1–8.

[18] CICA/AICPA, "Continuous auditing," 1999.

[19] C. E. Brown, J. A. Wong, and A. A. Baldwin, "A review and analysis of the existing research streams in continuous auditing," *J. Emerging Technol. Account.*, vol. 4, no. 1, pp. 1–28, 2007.

[20] M. A. Vasarhelyi, M. Alles, S. Kuenkaikaew, and J. Littley, "The acceptance and adoption of continuous auditing by internal auditors," *Methodol. J. AIS Res.*, vol. 13, no. 3, pp. 267–281, 2012.

[21] J. Woodroof and D. Searcy, "Continuous audit implications of internet technology," in *Proc. 34th Annu. Hawaii Int. Conf. Syst. Sci.*, Island of Maui, HI, US, 2001, pp. 1–8.

[22] K. Singh, P. J. Best, M. Bojilov, and C. Blunt, "Continuous auditing and continuous monitoring in ERP environments," *Inf. Syst. J.*, vol. 28, no. 1, pp. 287–310, 2013.

[23] J. R. Kuhn Jr, and S. G. Sutton, "Continuous auditing in ERP system environments," *Inf. Syst. J.*, vol. 24, no. 1, pp. 91–112, 2010.

[24] C.-C. Lin, F. Lin, and D. Liang, "An analysis of using state of the art technologies to implement real-time continuous assurance," in *Proc. 6th World Congress Serv.*, Miami, FL, USA, 2010, pp. 415–422.

[25] M. A. Vasarhelyi, M. G. Alles, A. Kogan, and D. O'Leary, "Principles of analytic monitoring for continuous assurance," *J. Emerging Technol. Accounting*, vol. 1, pp. 1–21, 2004.

[26] C.-H. Yeh, T.-P. Chang, and W.-C. Shen, "Developing continuous audit and integrating information technology in e-business," in *Proc. IEEE Asia-Pac. Serv. Comput. Conf.*, Yilan, Taiwan, 2008, pp. 1013–1018.

[27] U. S. Murthy and S. M. Groomer, "A continuous auditing web services model for XML-based accounting systems," *Int. J. Account. Inform. Syst.*, vol. 5, no. 2, pp. 139–163, 2004.

[28] S. Lamparter, S. Luckner, and S. Mutschler, "Formal specification of web service contracts for automated contracting and monitoring," in *Proc. Hawaii Int. Conf. Syst. Sci.*, Waikoloa, Hawaii, 2007, pp. 1–10.

[29] C. Ardagna, E. Damiani, R. Jhawar, and V. Piuri, "A model-based approach to reliability certification of services," in *Proc. 6th IEEE Int. Conf. Digital Ecosyst. Technol.*, Italy, 2012, pp. 1–6.

[30] P. Stephanow, C. Banse, and J. Schütte, "Generating threat profiles for cloud service certification systems," in *Proc. 17th IEEE High Assurance Syst. Eng. Symp.*, 2016, pp. 1–8.

[31] P. Stephanow and M. Gall, "Language classes for cloud service certification systems," in *IEEE 11th World Congress Serv.*, 2015, pp. 127–134.

[32] R. Ko, P. Jagadpramana, M. Mowbray, S. Pearson, M. Kirchberg, Q. Liang, and B. S. Lee, "TrustCloud," in *Proc. IEEE World Congress Serv.*, Washington, DC, USA, 2011, pp. 584–588.

[33] P. Mell, D. Waltermire, L. Feldman, H. Booth, A. Ouyang, Z. Ragland, and T. McBride, "CAESARS framework extension," 2012.

[34] M. D. Myers, *Qualitative Research in Business & Management*, 2nd ed. London, UK: SAGE, 2013.

[35] C. L.-Y. Chou, T. Du, and V. S. Lai, "Continuous auditing with a multi-agent system," *Decision Support Syst.*, vol. 42, no. 4, pp. 2274–2292, 2007.

[36] A. Ahmi and S. Kent, "The utilisation of generalized audit software by external auditors," *Managerial Audit. J.*, vol. 28, no. 2, pp. 88–113, 2012.

[37] T. Singleton and D. L. Flesher, "A 25 year retrospective on the IIA's SAC projects," *Managerial Audit. J.*, vol. 18, no. 1, pp. 39–53, 2003.

[38] N. Mahzan and A. Lymer, "Examining the adoption of computer-assisted audit tools and techniques," *Managerial Audit. J.*, vol. 29, no. 4, pp. 327–349, 2014.

[39] I. Pedrosa and C. J. Costa, "New trends on CAATTs," in *Proc. Int. Conf. Inform. Syst. Des. Commun.*, Lisboa, Portugal, 2014, pp. 138–142.

[40] J. Gao, "Technical framework model of continuous online assurance," in *Proc. Int. Conf. E-Business E-Government*, Guangzhou, China, 2010, pp. 2141–2144.

[41] G. Koschorreck, "Automated audit of compliance and security controls," in *Proc. 6th Int. Conf. IT Security Incident Manage. IT Forensics*, Stuttgart, Germany, 2011, pp. 137–148.

[42] M. Alles, G. Brennan, A. Kogan, and M. A. Vasarhelyi, "Continuous monitoring of business process controls," *Int. J. Account. Inform. Syst.*, vol. 7, no. 2, pp. 137–161, 2006.

[43] Y. Chen, "Continuous auditing using a strategic-systems approach," *Internal Auditing*, vol. 19, no. 3, pp. 31–36, 2004.

[44] B. Schroeder, "On-line monitoring," *Computer*, vol. 28, no. 6, pp. 72–78, 1995.

[45] Z. Rezaee, A. Sharbatoghlie, R. Elam, and P. L. McMickle, "Continuous auditing," *Auditing*, vol. 21, no. 1, pp. 147–163, 2002.

[46] J. E. Hunton and J. M. Rose, "21st Century Auditing," *Account. Horizons*, vol. 24, no. 2, pp. 297–312, 2010.

[47] R. L. Braun and H. E. Davis, "Computer-assisted audit tools and techniques: analysis and perspectives," *Managerial Audit. J.*, vol. 18, no. 9, pp. 725–731, 2003.

[48] A. Fuggetta, G. Picco, and G. Vigna, "Understanding code mobility," *IEEE Trans. Softw. Eng.*, vol. 24, no. 5, pp. 342–361, May 1998.

[49] J. M. Shaikh, "E-commerce impact," *Managerial Audit. J.*, vol. 20, no. 4, pp. 408–421, 2005.

[50] T. C. Du, E. Y. Li, and E. Wei, "Mobile agents for a brokering service in the electronic marketplace," *Decision Support Syst.*, vol. 39, no. 3, pp. 371–383, 2005.

[51] H. Ye, J. Yang, and Y. Gan, "Research on continuous auditing based on multi-agent and web services," in *Proc. Int. Conf. Manage. e-Commerce e-Government*, Beijing, China, 2012, pp. 220–225.

[52] J. Zhang and C. Wan, "Securing continuous auditing in wireless network," in *Proc. Int. Conf. E-Business E-Government* , Shanghai, China, 2011, pp. 1–4.

[53] C.-L. Fang, D. Liang, F. Lin, C.-C. Lin, and W.-C. Chu, "A portable interceptor mechanism on SOAP for continuous audit," in *Proc. Asia Pac. Softw. Eng. Conf.*, Bangalore, India, 2006, pp. 95–104.

[54] D. Żmuda, M. Psiuk, and K. Zieliński, "Dynamic monitoring framework for the SOA execution environment," *Proc. Comput. Sci.*, pp. 125–133, 2010.

[55] C.-T. Kuo, H.-M. Ruan, C.-L. Lei, and S.-J. Chen, "A mechanism on risk analysis of information security with dynamic assessment," in *Proc. 3rd Int. Conf. Intell. Netw. Collaborative Syst.*, Fukuoka, Japan, 2011, pp. 643–646.

[56] M. Jans, M. Alles, and M. Vasarhelyi, "The case for process mining in auditing," *Methodol. AIS Res.*, vol. 14, no. 1, pp. 1–20, 2013.

[57] Q. Fu, J.-G. Lou, Y. Wang, and J. Li, "Execution anomaly detection in distributed systems through unstructured log analysis," in *Proc. Int. Conf. Data Mining*, Miami, FL, USA, 2009, pp. 149–158.

[58] H. Du and S. Roohani, "Meeting challenges and expectations of continuous auditing in the context of independent audits of financial statements," *Int. J. Audit.*, vol. 11, no. 2, pp. 133–146, 2007.

[59] J. L. Perols and U. S. Murthy, "Information fusion in continuous assurance," *Inf. Syst. J.*, vol. 26, no. 2, pp. 35–52, 2012.

[60] C.-H. Wu, Y. E. Shao, B.-Y. Ho, and T.-Y. Chang, "On an agent-based architecture for collaborative continuous auditing," in *Proc. 12th Int. Conf. Comput. Supported Cooperative Work Des.*, Xi'an, China, 2008, pp. 355–360.

[61] Z. M. Jiang, A. Hassan, P. Flora, and G. Hamann, "Abstracting execution logs to execution events for enterprise applications," in *Proc. 8th Int. Conf. Quality Softw.*, Oxford, England, 2008, pp. 181–186.

[62] R. Accorsi, "Automated privacy audits to complement the notion of control for identity management," 2007.
[63] R. Accorsi and T. Stocker, "Automated privacy audits based on pruning of log data," in *Proc. 12th Conf. Enterprise Distrib. Object Comput.*, Munich, Germany, 2008, pp. 175–182.
[64] R. Nithiavathy, "Data integrity and data dynamics with secure storage service in cloud," in *Proc. Int. Conf. Pattern Recog., Inform. Mobile Eng.*, Salem, Germany, 2013, pp. 125–130.
[65] K. Yang and X. Jia, "An efficient and secure dynamic auditing protocol for data storage in cloud computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 9, pp. 1717–1726, Sep. 2013.
[66] Y. Zhu, G.-J. Ahn, H. Hu, S. Yau, H. An, and C.-J. Hu, "Dynamic audit services for outsourced storages in clouds," *IEEE Trans. Serv. Comput.*, vol. 6, no. 2, pp. 227–238, Apr.–Jun. 2013.
[67] C. Liu, J. Chen, L. Yang, X. Zhang, C. Yang, R. Ranjan, and K. Ramamohanarao, "Authorized public auditing of dynamic big data storage on cloud with efficient verifiable fine-grained updates," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 9, pp. 2234–2244, Sep. 2014.
[68] B. Wang, B. Li, and H. Li, "Knox" in *Proc. Appl. Cryptograph. Netw. Security*, 2012, pp. 507–525.
[69] H. C. H. Chen and P. P. C. Lee, "Enabling data integrity protection in regenerating-coding-based cloud storage," *IEEE Trans. Parallel Distrib. Syst*, vol. 25, no. 2, pp. 407–416, Feb. 2014.
[70] National Institute of Standards and Technology, NIST Cloud Computing Forensic Science Challenges: *Draft NISTIR 8006*.
[71] A. Pichan, M. Lazarescu, and S. T. Soh, "Cloud forensics," *Digital Investigation*, vol. 13, pp. 38–57, 2015.
[72] T. Kunz, P. Niehues, and U. Waldmann, "Technische unterstützung von audits bei cloud-betreibern," *DuD*, vol. 37, no. 8, pp. 521–525, 2013.
[73] J. R. Rajalakshmi, M. Rathinraj, and M. Braveen, "Anonymizing log management process for secure logging in the cloud," in *Proc. Int. Conf. Circuit, Power Comput. Technol.*, India, 2014, pp. 1559–1564.
[74] S. Zawoad, A. K. Dutta, and R. Hasan, "SecLaaS," in *Proc. 8th ACM SIGSAC Symp. Inform., Comput. Commun. Security*, Hangzhou, China, 2013, pp. 219–230.
[75] C.-H. Lin, C.-Y. Lee, and T.-W. Wu, "A cloud-aided RSA signature scheme for sealing and storing the digital evidences in computer forensics," *Int. J. Security Its Appl.*, no. 2, p. 241, 2012.
[76] National Institute of Standards and Technology, Federal Information Security Management Act of 2002.
[77] M. G. Alles, A. Kogan, and M. A. Vasarhelyi, "Audit automation for implementing continuous auditing," 2008.
[78] M. Becker, S. Lehrig, and S. Becker, "Systematically deriving quality metrics for cloud computing systems," in *Proc. 6th ACM/SPEC Int. Conf. Perform. Eng.*, Austin, TX, USA, 2015, pp. 169–174.
[79] S. Lins, P. Grochol, S. Schneider, and A. Sunyaev, "Dynamic certification of cloud services: Trust, but verify!," in *Proc. IEEE Security and Privacy*, vol. 14, no. 2, forthcoming, 2016.

Sebastian Lins is a research assistant at the Department of Information Systems, University of Cologne, Germany. His main interests in the field of information systems research are the (dynamic) certification and continuous auditing of cloud services.

Stephan Schneider is a postdoctoral researcher at the Department of Information Systems, University of Cologne, Germany. His research focuses on strategic decision-making in cloud sourcing projects as well as on cloud security and certification of cloud computing infrastructures.

Ali Sunyaev is an assistant professor at the Department of Information Systems, University of Cologne, Germany. He has published several international journal articles in leading journals such as *Communications of the ACM*, *Journal of Information Technology and IEEE Software*.

A Cloud-aided RSA Signature Scheme for Sealing and Storing the Digital Evidences in Computer Forensics

Chu-Hsing Lin[1], Chen-Yu Lee[2] and Tang-Wei Wu[1]

[1]*Department of Computer Science, Tunghai University, 407 Taichung, Taiwan*
[2]*Department of Computer Science, National Chiao-Tung University*
1001 Ta-Hsueh Road, HsinChu, 30050, Taiwan
{chlin, g99350031}@thu.edu.tw, chenyu@cs.nctu.edu.tw

Abstract

The privacy of data or plain is an important issue in cloud computing. Therefore, to solve the problem, we establish the environment of cloud computing to process data with privacy and apply our scheme in the area of digital forensics for RSA signature algorithm. We experiment with efficiency of RSA signature in cloud computing. As a result, our scheme can reduce the loading of computing; besides, the clients don't need to waste storage spaces to save the results. The most important of all, we can take full advantage of the cloud computing for computing of large data and storage spaces.

Keywords: *Cloud computing, Digital forensics, RSA signature, Privacy*

1. Introduction

Recently, cloud computing have been a new fashion noun in Information Technology, and the data security have become a new issue in the cloud computing [1, 2, 3]. Most mobile devices, such as cell phone, can't process large size data; they depend on the cloud services with rich resources.

The storage of sealed digital evidences of crime forensics [4, 5, 6] would change from traditional photo, video types stored in a specific building to cloud services via Internet. The powerful computation and high speed communication cloud services will speed up the sealing process, logistical forensics engineering. However, it would be unsecure at the first part of the scenario, the digital evidences sent from the mobile devices at the first line to the cloud service. The evidence would be sent in plain text, without encryption to the cloud service through the Internet, but it would suffer from eavesdropping by any malicious people. If the evidences need to be sent under encryption, nevertheless the mobile devices would have insufficient computation power to perform the necessary actions.

In the paper, we improved a secure protocol model [7] to propose a novel scheme that achieves the requirements needed in the mentioned scenario to make sure the security of sealing and storing the digital evidences from first line people to the cloud services. The rest of the paper is summarized as follows. Section 2 enhances the protocol to be better in burden of computing and storage. Section 3, we will analyze the results of experiment between tradition and cloud structure. Finally, conclusions are given in Section 4.

2. The Proposed Structure in Forensics

In this section, we described the steps of uploading the digital evidences to the data center of forensics with privacy and downloading for verification. The cloud data center of forensics are divided into two services, cloud computing center and cloud storage center and shown in Figure 1. Suppose that the forensics officers want to store the digital evidences *DE* on the forensics data center in the cloud. The stored digital evidence should be encrypted by the aid of the cloud server as $C = DE^d \bmod n$, where $n = p * q$, and (p, q) are two distinct prime numbers [8][9][10].

In the cloud computing center, the cloud data center couldn't know what the forensics officers upload, and still finish the work forensics officers want to do. On the other hand, forensics officers permute the *DE* by using the low computing functions and send it to the computing center. After the encryption at the computation center, it is than sent to storage center to seal up for keeping. The detailed steps are described as follows.

Forensics officers' part:

Step1: Generate t random numbers $a_1, a_2, \ldots, a_t$ such that p and q are not divisors of a_i and compute $a_0 = (DE\prod_{i=1}^{r} a_i) \bmod n$, for some $r < t$.

Step2: Compute $b_i = a_i^2 \bmod n$, for $i = 0,1,2,\ldots,t$ and compose the results to a vector $B = G(DE, (a_0, a_1, ..., a_t)) = (b_0, b_1, \ldots, b_t)$. Last, forensics officers send the result B to the cloud computing center.

Cloud computing center:

Step3: On receiving *B*, the cloud computing center permute on *B* by using random permutation ψ. Let $B' = \psi(B) = (b_0', b_1', \ldots, b_t')$.

Step4: Compute a vector $V = F(B') = (v_0, v_1, \ldots, v_t)$, where $v_i = (b_i')^{(d-1)/2} \bmod n$. Finally, send the signature *V* to cloud storage center.

Cloud storage center:

Step5: On receiving *V*, the cloud storage center compute $U = \psi^{-1}(V) = (u_0, u_1, \ldots, u_t)$ where ψ^{-1} is the inverse permutation of ψ. Note that $U = F(B)$.

Step6: Compute $C = (u_0 a_0)\left((\prod_{i=1}^{r} u_i)(\prod_{i=1}^{r} a_i)\right)^{-1} \bmod n = DE^d \bmod n$. Finally, store the result *C* in this center.

Verifier:

Step7: Download the result *C* from cloud storage center, and verify the *C* by *e* associated public key to obtain the result is $DE = C^e \bmod n$.

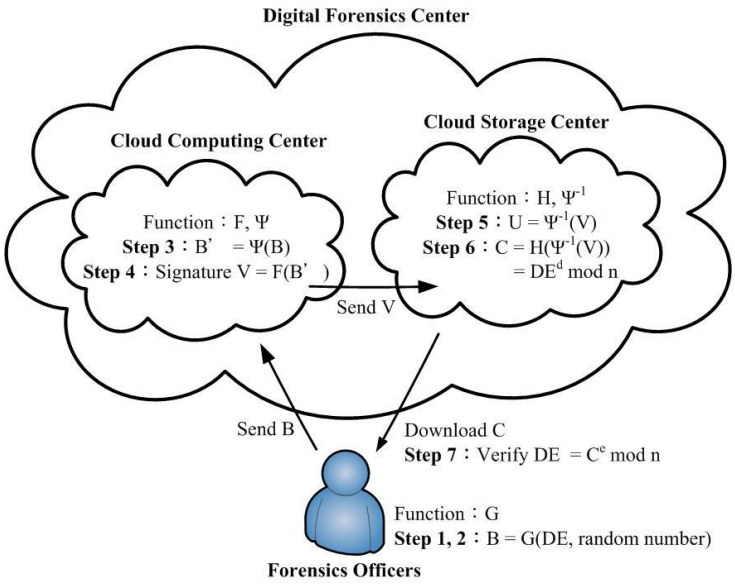


Figure 1. Steps of Cloud Structure

3. Experimental Results and Analysis

We would like to compare the efficiency of computation with the cloud structure and the traditional structure on the same experimental experiment. Among them, we set the argument of RSA key length 1024, 2048 and 4096, the number of random number is 40, and run 100 times to obtain the average time.

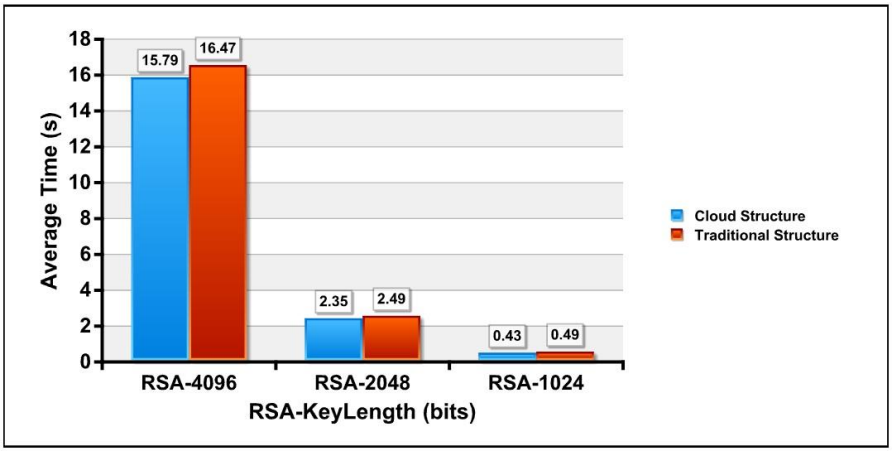


Fig. 2. Average Time with Different Key Length by Running 100 Times

As we can see in Figure 2, the results of the cloud structure represent a better efficiency than the traditional structure. The RSA-4096 in cloud structure is faster than the traditional structure about 0.68s, the RSA-2048 is about 0.14s, and RSA-1024 is about 0.06s. We find that if the key length is longer, and the execution time is lower. We can apply the cloud structure to some areas, such as digital forensics that we proposed in Section 2.

4. Conclusion

In this research, we propose a new digital forensics structure for RSA signature in cloud computing. This cloud structure can archive privacy, save computing power on mobile devices token by forensics officers. By RSA signature protocol, the verifier can verify the evidences in the court. Moreover, this protocol could be applied to many areas, such as digital forensics, online voting, or E-commercial, etc. We hope that we can accomplish online voting system in the future, and let it be used widely by people.

Acknowledgments

This work was supported in part by Taiwan National Science Council under grant: NSC 99-2221-E-029-039 -MY3.

References

[1] S. Subashini and V. Kavitha, "A survey on security issues in service delivery models of cloud computing", Journal of Network and Computer Applications, Vol. 34, pp. 1-11 **(2011)**.

[2] S. Ramgovind, M. M. Eloff and E. Smith, "The Management of Security in Cloud Computing", Information Security for South Africa (ISSA), pp. 1-7 **(2010)**.

[3] A. F. Mohammad and H. Mcheick, "Cloud Services Testing: An Understanding", Procedia Computer Science, Vol. 5, pp. 513-520 **(2011)**.

[4] S. J. Wang and D. Y. Kao, "Internet forensics on the basis of evidence gathering with Peep attacks", Computer Standards & Interfaces, Vol. 29, pp. 423-429 **(2007)**.

[5] Y. S. Yen, I. L. Lin and B. L. Wu, "A study on the forensic mechanisms of VoIP attacks: Analysis and digital evidence", Digital Investigation, Vol. 8, pp. 56-67 **(2011)**.

[6] F. Buchholz and E. Spafford, "On the role of file system metadata in digital forensics", Digital Investigation, Vol. 1, pp. 298-309 **(2004)**.

[7] C. H. Lin and C. C. Chang, "A Server-Aided Computation Protocol for RSA Enciphering Algorithm", Intern. J. Computer Math., Vol. 53, pp. 149-155 **(1993)**.

[8] R. Rivest, A. Shamir and L. Adleman, "A Method for Obtaining Digital Signature and Public Key Cryptosystems", Communications of the ACM, Vol. 21, No. 2, pp. 120-126 **(1978)**.

[9] S. Kawamura and A. Shimbo, "Performance Analysis of Server-Aided Secret Computation Protocols for the RSA Cryptosystem", The Transactions of The Institute of Electronics, Information and Communication Engineers IEICE, Vol. E73, No. 7, pp. 1073-1080 **(1990)**.

[10] F. Bao, C. C. Lee and M. S. Hwang, "Cryptanalysis and improvement on batch verifying multiple RSA digital signatures", Applied Mathematics and Computation, Vol. 172, pp. 1195-1200 **(2006)**.

A Comprehensive analysis of XML and JSON web technologies

Zia Ul Haq[1], Gul Faraz Khan[2], Tazar Hussain[3]

[1]*Management Information System Department, College of Business Administration, King Saud University, P.O. Box 71115, Riyadh 11587, Saudi Arabia*
[2, 3]*Department of Software Engineering College of Computer and Information Sciences, King Saud University, P.O. Box 51178, Riyadh 11543, Saudi Arabia*

[1]zparacha@ksu.edu.sa
[2]gfaraz@ksu.edu.sa
[3]tazhussain@ksu.edu.sa

Abstract—In this global era, internet plays a vital role to share information all over the world. There are some standard protocols and web technologies to represent the information on internet like HTML, JavaScript, ASP, JSP, XML, JSON etc. All these standards have pros and cons, and also depend on the requirement that which exchange of format more reliable. This research work shows the comparative analysis of XML and JSON using Multi-criteria Decision Support System MCDS, and analyze use of web technology according to the requirement need.

Keywords: **Web technology, JSON, XML, MCDS, AJAX**

I.INTRODUCTION

Today web technology is on the level that we are talking about synchronous and asynchronous transformation technologies like XML, AJAX, .NET, JSON and so on are in practice. But 15-16 years ago the people was only thinking how to display documents on the internet and for that purpose HTML (Hyper Text Markup Language) had been developed. And then with the time as people get use to with the internet, then the user start demands for new things like to have interactive websites that they can display information as well as can take input from user for that purpose different technologies had been developed that we are using in dynamic websites such that ASP, JSP, PHP, ASP.NET etc. then there was a problem how to transfer data between different platforms so XML has been developed which store plain text base data independent of platform and can easily transfer data between cross platforms.

In the last couple of years AJAX came in to existence which is totally new technology, in the same time JSON has also been introduced, and it is nowadays one of the most debatable topic for developers that "JSON is a fate free alternative to XML?" [10].

In this paper have discussed a little background of the web development starting from HTML, SGML, AJAX, XML and JSON and then XML and JSON is compared in different aspects and then on the basis of these comparisons conclusion has been derived that which technology is required for whom and for what purpose.

2. Background of XML and JSON

HTML is used and still the people are using it how to display some data or documents, how it will be look like and what will be its layout for that purpose, it uses some different tags like <HTML>, <HEAD>, <TITLE>, and <BODY> etc. to display documents. It was difficult to remember all the tags of HTML but as rapid developments are taking place in this field and now we can have different editors that we can use them how to design a HTML pages but still HTML has some limitations.

When the technology been more developed and the developers started thinking about to store and retrieve data to and from the database so HTML is ok with how to display data and documents but it is not suitable to deal with data structure like some stuff like database and objects hierarchies etc. [11].

2.1. SGML

As we know that HTML as good for displaying documents and their layout but HTML is not suitable for explicit quarries. So if we need to go for a bit different things like how to access and manipulate the data objects instead of documents then HTML is not capable of that, So SGML (Standard Generalized Markup Language). As we know that HTML is one of the applications of SGML. SGML is known as the mother tongue of HTML. So to overcome the limitation of HTML, SGML has the capability to overcome the problems of HTML many organizations had defined their own standards using SGML but SGML is very complex as its specifications are about 500 pages. Due to the very complex structure of SGML it turn the researcher to focus on such a tool which should overcome the limitations of HTML and should be

simple where the user can use in their application on the basis of simple standard and XML came into being [11].

2.2. AJAX (Asynchronous JavaScript and XML)

AJAX is tremendous development on the web which enables web application to look like desktop application means like in the normal WebPages when we need a bit of information we need to reload the whole page, but with AJAX it is now possible to transfer information with out reloading or refreshing the page. For example Google Suggest Lab or yahoo search when we are typing the word to search it give us suggestion with out reloading the page. It means that the AJAX technique enable pages to have the capability to transfer a small amount of data and can retrieve the data from the server in the same page, with out reloading the whole page. Now when we transferring this data we need data transfer format either XML or JSON can be used is data transfer formats. In the next section I am giving an overview to XML and then JSON and then I will come to give detail about which one XML or JSON to use with AJAX [7].

2.3. XML (Extensible Markup Language)

XML is a simple standard that can be used to transform and encode both text and data and it can be processed and transformed across different platforms. XML is more simple and manageable as compared to SGML, it is not a full independent markup language but it is a tool that enables users to define their own markup language, user can define their own tags which are more easy and readable by both machine, as well as user [11].

To take care of compatibility the XML is built to in the way to fit the HTML in the new framework and after some time HTML 4.0 which is a very suitable version that is compatible with XML and there it is named as XHTML, and simply we can say that it is nothing more but a special application of XML.

XML is a Meta data and the basic difference between HTML and XML is that, XML allow the user to define their own tags as HTML have a pre defined set of tags but in XML the user can define the tags according to their own use. And the other thing is that, that there are no formal criteria like the HTML have (head, title, body etc), it also take care of contents as well as XML do not defined any contents criteria but as XML is meta language so it allows contents base structuring as well as XLM is functioning different then the level HTML do [11].

So XML will need to be fid with XML based data by the user to get succeed. User can define their own tags according to their own need in their native language. Such that “StartDate”, “TrouserSize”, “EndTime” so XML do not define these tags but actually XML give the way how to define these tags, it means that XML have tags that apply to all documents (XML processing instruction) and also user can define their own criteria that can be stored as a separate documents (DTD) document type definition (Jung 2000). XML provide some grammar rules that apply to all documents, such that “<identifier>contents</identifier>” then the user can put the contents according to their own use for example <name>khan</khan>. Then everyone can use it according to their own use for instance automobile company can use it <EngineSize>, <ModelOfCar>, <Color> and then can save them in a special schema. The XML capable application can use these definitions directly.

Beside these as XML don’t understand these tags so it means that instead of <StartDate> if say you put <ABC123> XML parser will accept it without any problems it means that it could be anything that is understandable so for this purpose a group of user can be agree on some definitions of tags Document Type Definition (DTDs) can be used then [11].

XML architecture is very flexible documents are written in plain text not in cryptic form so it will be human readable and XML document can be stored, processed and distributed except any special DTD or XML code of the particular author [11].

As XML is flexible so any kind of tags can be defined. But DTD should be use for particular type of data such that real store, Publishing House etc. some of the organization and association already reached on agreement of such type of XML Documents definitions [11].

A simple example of XML data as given below,

```
<teaminfo>
  <players>
    <player>
      <name>Ajmal Khan</name>
      <height>5.8</height>
      <age>24</age>
      <postgrad>true</postgrad>
    </player>
    <player>
      <name>Nadeem shehzad</name>
      <height>5.9</height>
      <age>26</age>
      <postgrad>true</postgrad>
    </player>
        <player>
      <name>Aditya</name>
      <height>6.0</height>
      <age>24</age>
      <postgrad>true</postgrad>
    </player>
  </players>
</teaminfo>
```

The above XML example store information about three players in a team. We can see there are some elements that they are repeating for each peace of information how ever the actual data is repeated once. And we are interested in the players and their information. The elements <players> and <teaminfo> are not needed but they are just used to define the structure and meaning of the information. Table 1 shows advantages of XML technology over JSON [3].

Table 1. XML strengths over JSON

JSON	XML
There is no grammar support and that's why it is difficult to communicate and enforce interface contracts	While XML have XML schema and Document Type Definition which can be used to define grammar rules
Extensibility is not good as namespaces are not supported	Very strong support for namespaces, schema have more extensibility options
Development tools support is very limited as it is newly introduced	As XML is in the market since long time there for is supported by most of the development tools
JSON is very narrow focused as it is used only for Remote Process Call(RPC), mainly with JavaScript Clint	XML is very broad focused, it can be used for Remote Process Call (RPC), Electronic Data Interchange (EDI), Metadata etc.
Very limited support for web services associated stuff (products).	huge hold of web services related products

2.4. JavaScript Object Notation (JSON)

As web services are gaining attractiveness day by day, XML has almost turned into the actual paradigm for data transmission. But still there are different things that people considering that XML is heavy some time it send more bytes through the internet to get the things done which can be done with a much smaller data. To overcome this problem new formats of XML been introduced like binary XML so it means all these solutions are extending XML but still the problem exist when it come to backwards compatibility. Douglas Crockford is software engineer he introduced a new data format which is based on JavaScript called JavaScript Object Notation (JSON) [6].

JSON is simple very lightweight object serialization technique or data format which is based on JavaScript Object initialization syntax, specifically array and object literals. JSON definitions can be incorporated inside JavaScript files and accessed with no further parsing that comes alongside with XML-based languages, because it uses JavaScript syntax. But prior to use JSON, it is essential to know the array and object literals particular JavaScript syntax. JSON the initialization code is assigned to a string and then is dealt with JavaScript eval() function or JSON parser [6].

The JSON parser is very light weight. JSON is mainly used with different AJAX tools kits and frameworks and provide easy serialization for remote calls. JSON is supported by GWT and DOJO. JSON and web 2.0 technologies must be considered very seriously by Service Oriented Architecture (SOA) (Alexander 2007).

Most of the server languages do not contain JavaScript interpreters there for they wouldn't able to process the JavaScript code and lets suppose that they can evaluate it, but still the developer wouldn't allow the arbitrary code to b run on the server because it can generate a serious security problem. But both of these problems been solved by JSON which is the literal Syntax for JavaScript, as JSON can be run using eval() function of the JavaScript side. JSON allows any JavaScript data types to be transferred and would be faster than the XML-based solutions because the compact encoding allows for much smaller data to be transferred. On server side small parser to be built to serialize the native data types into JSON and also to create native data types from JSON [7]. The following sample describes how to represent JSON data.

```
{ "teaminfo" :
  {
    "players" : [
      {
        "name" : "Ajmal Khan",
        "height" : 5.7,
        "age" : 24,
        "postgrad" : true
      },
      {
        "name" : "nadeem shehzad",
        "height" : 5.8,
        "age" : 26,
        "postgrad" : true
      },
      {
        "name" : "Aditya",
        "height" : 6.0,
        "age" : 23,
        "postgrad" : false
      }
    ]
  }
}
```

ISBN: 978-1-61804-285-9

We can observe that a lot of redundant information is not present as compare to the XML one; no closing tags are required to match opening tags this will reduce the number of bytes to be send out for same information of to great extent. In the above example excluding spaces the JSON data is 249 bytes while the XML data is 378 bytes so it save more than 120 bytes in this much data. That is the reason why Crockford, JSON inventor said that "JSON is a fat free alternative to XML".

The draw back of the JSON format is that it is a bit hard to read as compare to XML as XML one is easier to read by a layperson as it is clearer and meaning full but JSON format is reduced by its shorthanded notations and due to that it will be difficult to read it with nicked eye. But no it can be disagreement that why we need to view the data exchange format with naked eye if we can use tools for parsing the data passing back and forth but still the question arising that are such tools available so there are some tools available but still this can be a limitation some where (EICHORN 2006). JSON have some pros over XML as shown in table 2 [3].

Table 2. JSON strengths over XML

JSON	XML
Completely programmed technique for de-serializing and serializing JavaScript objects, with very little coding.	JavaScript code will be written by developer to serialize and de-serialize to and form XML
Most of the browsers have enough support of JSON.	All new browsers have built-in XML parser but it could be a bit tricky when it come to cross-browser XML parsing.
The format is very concise due to having name/value pair-based approach.	Because of tags and namespaces the format is very lengthy.
de-serialization is very speedy in JavaScript	De-serialization is slower in JavaScript
Most of JavaScript libraries and AJAX toolkits have good support of JSON	AJAX toolkits don't have strong support for it.
Having simple API for JS and more other languages	The APIs are very complicated

3. Related work

Data interchange format have significant consequences on data transferring rates and performance, data interchange format generate from mark-up to further support for structural attribute of information using encoding of meta-data. XML and JSON are data interchange format that can be use in different aspects with unique purpose. XML primary uses are object serialization for transfer of data between application and Remote Procedure Calls RPC [13]. To analyze the impact of management of energy and cost of processing for transferring data in mobile devices of proposed formats of XML, JSON and Protocol Buffer [5]. These two form of data interchangeable using data serialization approach which allows for better communication between applications. Data transmission of web application more secure, powerful in the XML serialization approach and with JSON serialization approach fast and convenient [17].

XML is more complex than JSON in web services of web programming, some time programmer doesn't need to use namespace and mixed content documents. The developers focus on to use simple data structure, compact and exchange format. XML is great in problem domain, namespace, well-formed and mixed content document [15].

The Analytic Hierarchy Process is a method of measurement for formulating and analyzing decisions. AHP is a decision support tool which can be used to solve complex decision problems considering tangible and intangible aspects. Therefore, it supports decision makers to make decisions involving their experience, knowledge and intuition [2],[9].

4. Comparative analysis

In this research work comparison between JSON and XML carried out by using the Make it Rational MCDM tool. MCDM is decision making tool based on comparative analysis and defined steps. For decision making system these are the main steps to be evaluate goal, alternative, criteria, preferences, sub criteria and final result [1].

4.1. Goal

Main goal for this work to be analyzed are JSON and XML technologies based on criteria and sub criteria. This work will help developer to choose proper web technology in certain condition [12].

4.2. Alternatives

There are many web development technologies, this research work focus on alternatives of JSON and XML.

4.3. Main criteria

Hierarchy view show top down approach for alternatives, criteria's and sub criteria's, as shown in Figure 1, in our case main criteria's are Format of Exchange, Validity, Readability, and sub criteria's are Machine and Human Readability.

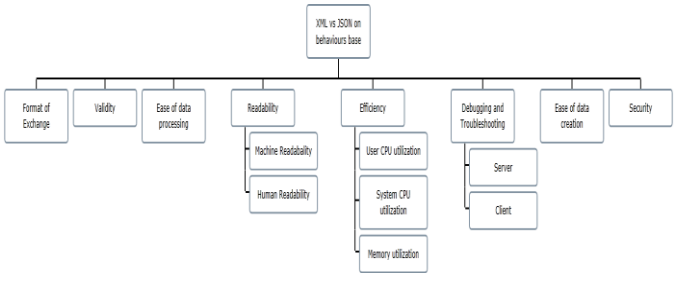


Figure 1: Hierarchy view of XML and JSON

4.3.1. Format of Exchange

JSON format is always smaller than XML, in fact the more tag involves in XML format increase size of XML exchange format. JSON specification excluding unnecessary tag and syntax produce small size of exchange format comparatively to XML. AJAX application can use XML or JSON as transformation format. Now it is an important issue how to select transformation format. XML is more composite structure and could be able to transfer any type of data however JSON is simple data structure that will be all we need to transfer AJAX data, and it is useful enough to use JSON with AJAX such that, JSON is subset of JavaScript, so using eval() method of JavaScript JSON text can be simply changed into JavaScript Object and then we can extract data using JavaScript. So if someone know JavaScript will be easy for them to use.

JSON is subset of JavaScript so it have the JavaScript Data types but it is not same for XML we can define it for XML using XML schema or DTD to define XML structure.

JSON can be parsed as JavaScript, for more security we can use the JSON parser to convert. But if we use to get the as XML it means that we will need to parse it. For that Dom method will be needed which is very comple [16].

4.3.2. Validity

XML content rules conforms the data to be valid documents. These rules describe document organizational structure and accurate data values. Valid XML documents match defined schema, constrains on the structure and content of the document articulated in schema. JSON validator is a program that verifies JSON data with provided schema which contains define validation, documentation, interaction control of JSON data and hyperlink navigation. Research analysis shows that XML validation bit stronger than JSON.

4.3.3. Ease of Data processing

The simple data structure and data standard of XML provide very easy process. But it is same in JSON as it having very simpler structure [10]**.**

4.3.4. Efficiency

XML document includes statements, handling instruction, elements and rich tags. Also composed of root node, the root node contains a root element, nested child element also include chilled element and properties which increase size of the XML documents. Data format of JSON is very simple that can be transmitted with a single array, variable of number or Boolean type or also string type. JSON exchanging data by object while object are tagged by unordered which contains a series of key values and pair key value [14]. Analysis and t-test observation indicated that XML appears to use less user CPU utilization than JSON, XML use more system CPU utilization than JSON, and memory utilization of the JSON and XML encoded transmissions nearly the same on server [13].

4.3.5. Debugging and Troubleshooting

XML server checks data being sent to client well formed and valid. XML document verify with XML schema, as an alternative to XML, JSON manually involves verification that the response object has the right attributes. On client side it is difficult to spot error in either format; browser would fail to parse XML into the response XML. For small data relatively easy to detect error in JSON but with large data it is difficult to relate the error message to the data.

4.3.6. Ease of Data creation

XML data-binding APIs to create XML in more than a few programming languages, XML APIs have been around for years and may be deal with complex application. JSON APIs are new to create JSON responses but not so far behind than XML. There are so many ways to create XML alternative to JSON [4]. Security

XML language specifications ensure that the form of serialization data of XML has strong security; labels stored all data in the tag closed strictly with the data index. AJAX lightweight application demanding for low security while have high demanding for efficiency, so JSON have good support as an alternative to XML [17].

4.3.7. Extensible

Extensibility reduces the coupling between the producer and the consumer of the data. XML is extensible while JSON is not but there is no need for that because it is not document mark-up language so there is no need for new tags as at already store data so there is no need to have tags to store data about data [10].

4.3.8. Reusability of software

As XML claim that the there are plenty of software code is available that developer can use that code and there is no need for recoding but JSON is simpler and there is no need for more programming/ additional software only JSON simple code is enough [10].

4.3.9. Adoptability by the industry

XML is adopted by the wide range of computer industry. Hover JSON is just newly known to the industry and because of it simplicity and it is easy to convert JSON from XML, due to these properties JSON becoming more adoptable [10].

4.4. Preferences

Preference is concerned with the priorities based on importance of criteria/sub-criteria, as assigned priorities are shown in the table 3.

Table 3. Preference based on priorities

Intensity	Importance	Intensity	Importance
1	Equal Importance	6	Strong Importance plus
2	Weak Importance	7	Very Strong Importance
3	Moderate Importance	8	Very Strong Importance plus
4	Moderate Importance plus	9	Extreme Importance
5	Strong Importance		

Each criteria assigned priority value shown in the given table 4.

Table 4. Criteria priority scale based on importance

Criteria	**Ratio**	**Criteria**	**Ratio**
Ease of data processing vs. Format of Exchange	1:1	Efficiency vs. Ease of data creation	3:1
Validity vs. Readability	3:1	Format of Exchange vs. Ease of data creation	2:1
Format of Exchange vs. Security	1:1	Security vs. Efficiency	1:1
Efficiency vs. Readability	2:1	Ease of data processing vs. Ease of data creation	2:1
Ease of data processing vs. Readability	2:1	Ease of data processing vs. Efficiency	1:1
Validity vs. Ease of data creation	3:1	Debugging and Troubleshooting vs. Efficiency	1:1
Security vs. Ease of data creation	2:1	Ease of data creation vs. Readability	1:1
Format of Exchange vs. Debugging and Troubleshooting	1:1	Security vs. Debugging and Troubleshooting	1:1
Debugging and Troubleshooting vs. Ease of data creation	2:1	Validity vs. Security	2:1
Validity vs. Efficiency	2:1	Format of Exchange vs. Efficiency	1:1
Debugging and Troubleshooting vs. Readability	3:1	Ease of data processing vs. Security	1:1
Format of Exchange vs. Readability	2:1	Validity vs. Ease of data processing	2:1
Ease of data processing vs. Debugging and Troubleshooting	1:1	Security vs. Readability	2:1
Validity vs. Format of Exchange	2:1	Validity vs. Debugging and Troubleshooting	2:1

4.5. Result and analysis

4.5.1. Ranking graph

Figure 2 shows rank graph for each criteria of XML and JSON, i.e. format of the exchange rank graph for JSON contribute more than XML, while the validity of an XML value higher than JSON.

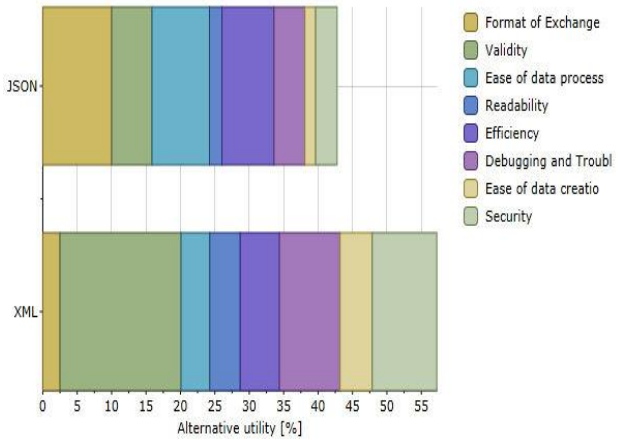


Figure 2: XML and JSON Attributes ranking graph

4.5.2. Values allocation table

Values assign to criteria's in alternatives of XML and JSON as shown in Table 3, i.e. format of exchange 2.5 value for XML and 10.01 value of JSON for same criteria.

Table 3: Values allocation to criteria of XML and JSON

Criteria	**XML**	**JSON**
Format of Exchange	2.5	10.01
Validity	17.6	5.87
Ease of data process	4.17	8.34
Readability	4.41	1.81

Efficiency	5.51	7.57
Debugging and Troubleshooting	8.8	4.48
Ease of data creation	4.67	1.56
Security	9.38	3.13
Total	57.23	42.77

4.5.3. Weight Chart

Criteria's Weight distribution shown in Figure 3, Validity has the highest value and lowest value for Ease of data creation.

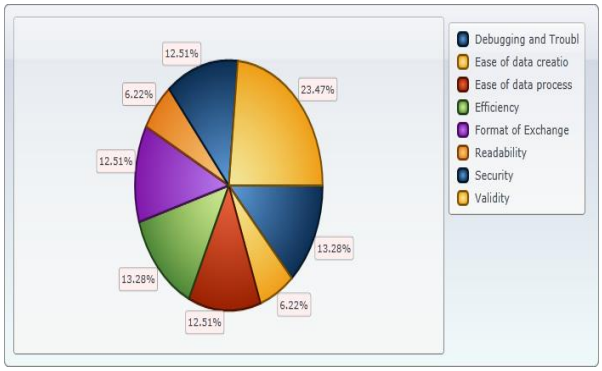


Figure 3: Weight chart of XML and JSON attributes

4.5.4. Alternative Chart of XML and JSON

In readability, security, validity and debugging and troubleshooting XML have edge over JSON, while JSON better in format of exchange, efficiency and ease of data processing than XML as shown in Figure 4.

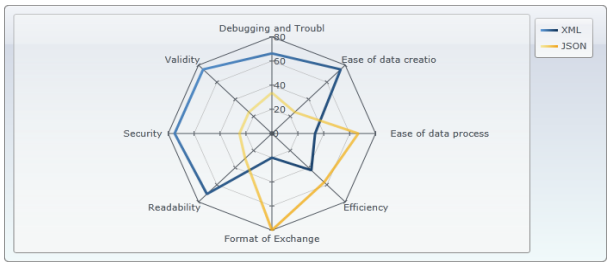


Figure 4: Alternatives chart of XML and JSON

4.5.5. Efficiency ranking graph

For each criterion in alternatives have ranking graph, as figure 5 show graph for efficiency further extended to sub criteria's of user CPU utilization, system CPU utilization and memory utilization. The graph shows that JSON best in user CPU utilization than XML, while in system CPU utilization XML perform well than JSON.

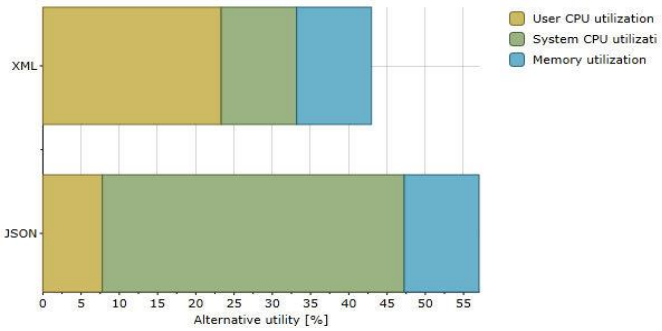


Figure 5: ranking graph for efficiency of XML and JSON

4.5.6. Efficiency Alternation chart

Alternative chart for efficiency in figure 6 shows that system CPU utilized well by alternative JSON, XML efficiently use user CPU and in memory utilization both alternative are same.

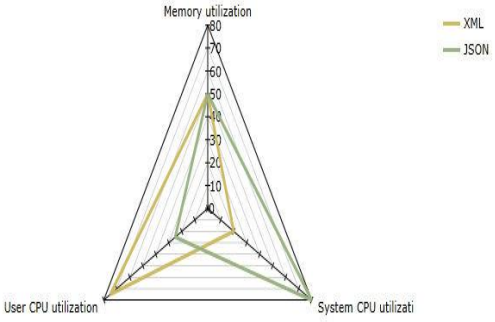


Figure 6: Alternative chart for attribute of XML and JSON Efficiency

5. Conclusion

From the above comparison it is clear that both technologies JSON and XML have their own advantages and drawbacks and it is also clear that both can be use according to the need of the system. The research study, understanding and facts from the above graphs and results conclude that XML have a bit edge over JSON for web technologies. According to my experience and above description I will advice that combination of both should be used depending on the requirement and demand. Both technologies have good properties for different situation as discussed above.

ACKNOWLEDGEMENT

This work was supported by the Research Centre of College of Computer and Information Sciences, King Saud University. The authors are grateful for this support.

References

[1] Abdullah S. Alghamdi, H. U., Syed Usman Ali (2011). Evaluating Chaos-based vs. Conventional Encryption Techniques for C4I System. International Conference on Computer Communications and Networks (ICCCN 2011). Lahaina, Hawaii, USA.

[2] Abdullah Sharaf Alghamdi, I. A., Muhammad Nasir (2010). Evaluating ESB for C4I Architecture Framework Using Analytic Hierarchy Process. Software Engineering Research and Practice Las Vegas, Nevada, USA

[3] Alexander (2007). "JSON Pros and Cons ". Retrieved April 25, 2012, from http://myarch.com/json-pros-and-cons.

[4] Allamaraju, S. (2006). "JSON vs XML." Retrieved May 28, 2012, from http://www.subbu.org/blog/2006/08/json-vs-xml.

[5] Bruno Gil, P. T. (2011). Impacts of data interchange formats on energy consumption and performance in smart phones. OSDOC '11 Proceedings of the 2011 Workshop on Open Source and Design of Communication, NY, USA ACM.

[6] C. Zakas, N. M., J. and Fawcett, J. (2006). Professional AJAX, University of Huddersfield.

[7] EICHORN, J. (2006). Understanding AJAX United States: prentice hall, University of Huddersfield.

[8] Esposito, D. (2007). Introducing Microsoft ASP.NET AJAX Microsoft Press, University of Huddersfield

[9] Iftikhar Ahmad, A. A., Abdullah Sharaf Alghamdi (2010). "Evaluating Intrusion Detection Approaches Using Multi-criteria Decision Making Technique." International Journal of Information Sciences & Computer Engineering (IJISCE) **1(1)**: 60-67.

[10] JSON (2005). "JSON: The Fate-Free Alternative to XML ". Retrieved April 10, 2012, from http://www.json.org/xml.html.

[11] Jung, F. (2000). "Backgrounder technology and application ". Retrieved February 06, 2008, from http://www.softwareag.com/xml/about/e-XML_Backgrounder_WP03E0700.pdf.

[12] Khalid Alnafjan, T. H., Gul Faraz Khan, Hanif Ullah, Abdullah Sharaf Alghamdi (2012). "Comparative Analysis and evaluation of Software Security Testing Techniques." International Archive of sciences journal.

[13] Nurzhan Nurseitov, M. P., Randall Reynolds, Clemente Izurieta (2009). Comparison of JSON and XML Data Interchange Formats: A Case Study. 22nd International Conference on Computer Applications in Industry and Engineering, Hilton San Francisco Fisherman's Wharf, San Francisco, California, USA, CAINE.

[14] Peng Wang, X. W., Huamin Yang (2011). Analysis of the Efficiency of Data Transmission Format Based on Ajax Applications. Information Technology, Computer Engineering and Management Sciences (ICM). Nanjing, Jiangsu, IEEE. **4:** 265 - 268.

[15] Sporny, M. (2010). "Web Services: JSON vs. XML." Retrieved June 02, 2012, from http://digitalbazaar.com/2010/11/22/json-vs-xml/.

[16]Ullman, C. a. D., L. (2007). Beginning AJAX Wrox press, University of Huddersfield

[17] Wang, G. (2011). Improving Data Transmission in Web Applications via the Translation between XML and JSON. Third International Conference on Communications and Mobile Computing. Qingdao, IEEE: 182 – 185

A Portable Interceptor Mechanism on SOAP for Continuous Audit

Chen-Liang Fang[1], Deron Liang[2], Fengyi Lin[3],
Chien-Cheng Lin[2], William Cheng-Chung Chu[4]
[1]Jin-Wen Inst. of Tech., [2] National Taiwan Ocean University, [3]Chihlee Inst. of Tech., [4]Tunghai University
Email: fang@jwit.edu.tw, drliang@iis.sinica.edu.tw, linfengyi.tw@yahoo.com.tw, hammerlin@hotmail.com

Abstract

Web Services has become popular in modern distributed applications, and the SOAP technology currently is the most used in Web Services. Recent middleware research works widely use interception approach for many problem domains, for example Fault Tolerance, Continuous Audit (CA), security etc. Instead of providing a Portable Interceptor as CORBA does, SOAP 1.2 provides an intermediary mechanism. Due to backward compatibility issue, we found intermediary mechanism is not a feasible solution for interception. Furthermore, our use case analysis found that CORBA Portable Interceptor functionality does not fulfill all requirements of Continuous Audit. This motivates us to develop a new Portable Interceptor Mechanism (PIM) on SOAP for CA. In this paper, we propose a PIM on SOAP to meet the interception requirements for Web Services. Our PIM includes portable interceptor management in SOAP engine and portable interceptor interface definitions.

1. Introduction

SOAP technology has several advantages over other middleware technologies when it comes to building a distributed application system [8]. Our previous SOAP related works [4][6] had shown that an interceptor mechanism is needed to fulfill the requirements on logging, client fault transparency, and redundant nested invocation problem in a FT system. Based on our recent survey, we notice that recent middleware research works use interception approach in many other problem domains. For example, FT-SOAP [6] uses interception approach in logging/recovery mechanism; [9], [18] uses an interceptor to audit an accounting information system; and dSniff 16 uses interception approach in computer forensics auditing.

Instead of providing portable interceptor, SOAP 1.2 provides intermediary mechanism. Based on the discussion in [8], a SOAP 1.1 client might not be able to route message to intermediary node by adding routing information, <via> tag, to the SOAP message. This is so called a backward compatibility issue. We found intermediary approach is not a feasible solution for interception due to backward compatibility issue. Many SOAP 1.2 compliant products provide ways that a request can be pre-processed or post-processed by a third-party application. Examples are SOAP Handler of Apache Axis 1.3[1] and WSE filter of Microsoft [12]. Using these proprietary interceptor mechanisms might cause portability problem.

We notice that SOAP and CORBA have many architectural technologies in common. For example, the services are defined in WSDL [20] in SOAP and IDL in CORBA [13]. On the other hand, they differ in many ways. For example, SOAP 1.2 doesn't support portable interceptor mechanism that CORBA does [13]. This motivates us to propose portable interceptor on SOAP by referring to Portable Interceptor of CORBA [13]. Based on our use case analysis, presented in Section 2, for the system requirements, the specification of CORBA Portable Interceptor is not enough to fulfill the requirements of some interception applications. For example, there are no administrative functions which are needed in Continuous Audit (CA) application.

In this paper, we propose a **portable interceptor mechanism** (PIM) on SOAP to meet the interception requirements for Web Services. Our PIM includes portable interceptor management in SOAP engine and portable interceptor interface definitions.

The remainder of this paper is structured as follows. Section 2 analyzes the PIM feature by use cases. In Section 3, we discuss our PIM design in

XIII ASIA PACIFIC SOFTWARE ENGINEERING CONFERENCE (APSEC'06)
0-7695-2685-3/06 $20.00 © 2006 IEEE

detail. The prototype implementation is given in Section 4 and then followed by Evaluation in Section 5. Our conclusions are detailed in Section 6.

2. The Functional Requirements of PIM

In this section, we would like to analyze the basic functional requirements of the proposed portable interceptor (PI) in SOAP. We analyze the functional requirements by use cases. The use cases not only show the requirements of portable interceptor in SOAP but also show why the proposed interceptor is completely compliant to current SOAP standard [21].

According to Liang's work on continuous audit (CA)[9], we suggest that the PIM has to fulfill the following requirements:

R1 Automated monitoring procedures will provide most of the audit evidence necessary to opine on the subject of the tertiary monitoring at real time or closed to real time opinion real-time or near real-time monitoring

R2 The monitoring mechanism and the platform should be easy to use.

R3 The subject of the monitor entity has suitable characteristics necessary to conduct the tertiary monitoring.

R4 The concurrent technique designed under continuous monitoring can be adjusted in seconds by those tertiary monitoring, with limited technical background.

R5 When a continuous monitoring system is not functional, it could not affect the reviewee's EDP systems

Manage a Portable Interceptor

The auditor uses an admAP to manage an auditing mechanism, implemented as a portable interceptor, as shown in Figure 1. The admAP invokes a plug-in service call in the SOAP engine. The SOAP engine finds the location (or URI) of the pluggable auditing mechanism (a portable interceptor) and loads (or plug) into the SOAP engine, as shown in the step 1 of Figure 1. After plugging this portable interceptor, the auditor sets the desired initial auditing parameters in the auditing interceptor shown in Step2. Then the auditor activates the plugged portable interceptor via the same admAP, as shown in the Step 3 of Figure 1. Furthermore, the administrator can deactivate the functions of the plugged portable interceptor by invoking a deactivation function to the SOAP engine, as shown in Step 4 of Figure 1. Furthermore, the auditor may alter runtime auditing options via SOAP engine service after the portable interceptor is plugged.

Based on the above use case, we find that the **plug-in** feature fulfills the requirement R2 because it can install interceptor by simply providing the location of the monitoring module; the **set property** feature supports Requirement R3 and R4; and the **activate/deactivate** feature support Requirement R4 and R5.

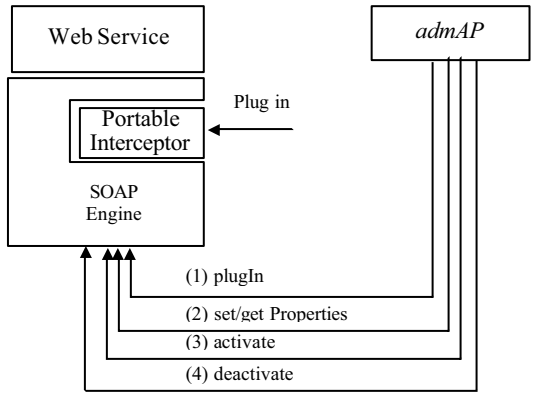


Figure 1. Interceptor management on Web Services server.

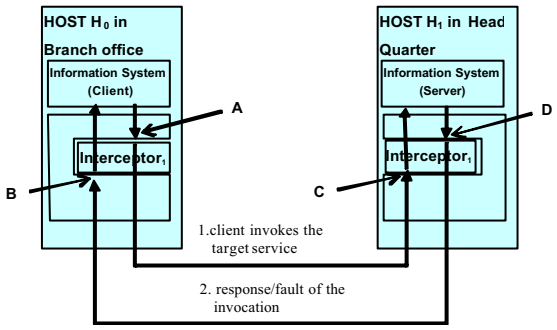


Figure 2. The four interception points of interceptors in a Web Services.

Runtime Phase of a Portable Interceptor

Suppose that a multi-national corporation use Web Services to integrate the information systems in branch office and head quarter. The information system in branch is a Web Services client and the one in head quarter is a Web Services server. The auditor may install the client side interceptor in the system located either in branch office or a server side

0-7695-2685-3/06 $20.00

interceptor in the system located in head quarter. The both client and server auditing interceptor intercepts transactions at two interception points. A Web Services client side interceptor can intercept outgoing requests (interception point A) and requested response/fault (interception point B) when the Web Services response. Furthermore, a server side interceptor can intercept arrival requests before the Web Services serves the request (interception point C) and/or intercept the response (interception point D) after the Web Services application return the response/fault to the client.

Based on the above use case, we observe that the portable interceptor is able to intercept all transactions in the system and fulfills the requirement R1. Table 1 summarizes what the requirements are fulfilled by theses features.

Feature \ Requirement	R1	R2	R3	R4	R5
Client (Server) side interceptor	X				X
Plug-in		X			
Activate/deactivate				X	X
Set/get property			X	X	

Table 1. The features of PIM fulfill the requirements.

3. The Portable Interceptor Design

Based on the use case analysis, we propose portable interceptor architecture to fulfill the requirements of portable interceptor. The requirements of portable interceptor are categorized into two folds: the portable administration features in SOAP engine; and a four-interception-point interception mechanism. We shall present the administration features design in SOAP engine first and then followed by the portable interceptor design in the following paragraphs.

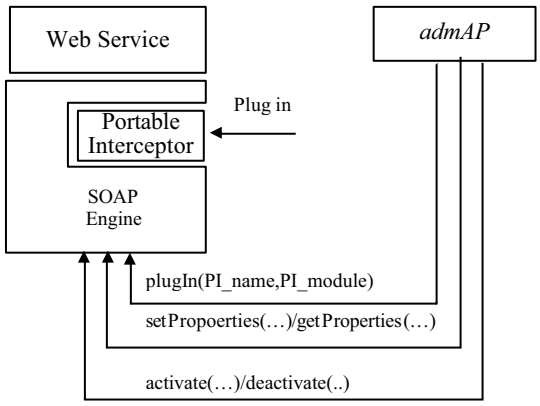


Figure 3. The portable interceptor management design: An usage example.

Portable Interceptor Administration API Design

As shown in Figure 3, the SOAP engine is implemented five portable interceptor administration functions. The Web Services system administrators can implement an administration application, admAp of Figure 3, to manage all portable interceptors by using the administration APIs. The four administration functions are named as *plugIn*(), *setProperties*(), *getProperties*(), *activate*(), and *deactivate*(). A portable interceptor is loaded(or plugged) into SOAP engine by using *plugIn*(*PI_name*, *PI_module*) function. The advantage of *plugIn*() design is no-interruption installation. The *plugIn*() provides a non-stop PI installation mechanism for a critical web service. As a result, the Web Services will not be interrupted during PI installation. The system administrator then invoke *setProperties*(*PI_name, Properties*) on the SOAP engine to setup all desired operation parameters. After setting all desired parameters for interception operation, the PI is able to be activated by calling *activate*(*PI_name*) on the SOAP engine. And vice versa, the installed PI can be deactivated by calling *deactivate*(*PI_name*) if the system administrator want to deactivated a specific PI at desired time.

Figure 4 depicts the excerpt of portable interceptor definition in WSDL format. The WSDL is not very readable for presentation purposes. Fortunately, OMG proposed a mapping specification between OMG's IDL and WSDL [14]. We convert

0-7695-2685-3/06 $20.00 © 2006 IEEE

WSDL of the PI manager into OMG IDL base on [14]. The excerpt of portable interceptor manager in IDL format is shown Figure 5. In the remainder of this paper, we present the portable interceptor interface using OMG IDL.

```
<?xml version="1.0" ?>
<definitions name="PI_Manager" …>
   <types>
     <schema …>
       <xsd:simpleType name="PI_module">…
       <xsd:simpleType name="PI_name"> …
     </schema>
  </types>
  <message name="plugIn">
    <part name="PI" type="xsd:PI_name" />
    <part name="pim" type="xsd:PI_module" />
  </message>…
  <message name="setPropertiess">
    <part name="props" type="xsd:Properties" />
  </message>…
  <message name="getProperties">
  </message>…
  <message name="activate">
    <part name="PI" type="xsd:PI_name" />
  </message>…
  <message name="deactivate">
    <part name="PI" type="xsd:PI_name" />
  </message>…
  <portType name="PI_ManagerPortType">
     <operation name="plugIn">…
     <operation name="setPropertiess">…
     <operation name="getProperties">…
     <operation name="activate">…
     <operation name="deactivate">…
  </portType>
  <binding …>
    …..
  </binding>
  <service name="PI_ManagerService">
     …
  </service>
</definitions>
```

Figure 4. The excerpt of portable interceptor manager interface definition (WSDL format)

```
typedef string PI_name;
typedef string PI_module;

interface PI_Manager {
   void plugIn(in PI_name PI, in PI_module pim);
   void setProperties(in PI_name PI, in Properties props);
   Properties getProperties(in PI_name PI);
   boolean activate(in PI_name PI);
   boolean deactivate(in PI_name PI);
}
```

Figure 5. The excerpt of interceptor manager interface definition (OMG IDL format)

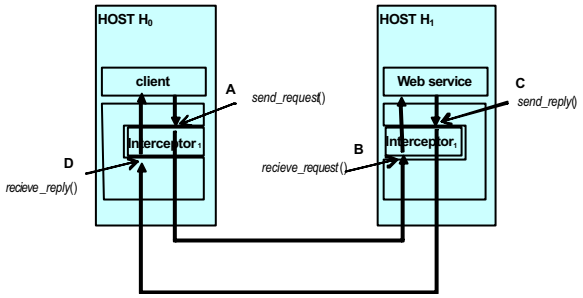


Figure 6. The API design of Portable Interceptor

Portable Interceptor API Design

The Figure 7 depicts our potable interceptor interface design. We define a basic interceptor, a client side interceptor, and a server side interceptor. The detail design of the three interceptor interfaces are presented in the following paragraph in order.

A basic interceptor interface, named **Interceptor**, is defined in Line 2 to Line 10 of Figure 7. The *Interceptor*::*initialize*(), Line 3 in Figure 7, is called by SOAP engine after loading the portable interceptor in initialization phase. The interceptor programmers may set up all necessary running environment parameters for all desired purposes. The *Interceptor*::*destroy*(), Line 4 in Figure 7, allows SOAP engine to clean up a unused portable interceptor for management purpose in terminal phase. The *Interceptor*::*setPropoerties*(*PI_name,Properties*) and *Interceptor*::*getPropoerties*(*PI_name*) allows administrator to set additional properties and query the running properties during runtime. Furthermore, the SOAP engine calls *Interceptor*::*activate*() to activate the interceptor and *Interceptor*::*deactivate*() to temporary deactivate the interceptor when the SOAP engine is invoked *PI_Manager*::*activate*() and *PI_Manager*::*activate*() by *admAP* when the interceptor administrator desires to do so. The basic interceptor is inherited by **ClientInterceptor** and **ServerInterceptor**. As a result, the **ClientInterceptor** and **ServerInterceptor** have these basic features.

```
1 : //IDL definitions of portable interceptor
2 : local interface Interceptor {
3 :     void initialize();
4 :     void destroy();
5 :     void setProperties(in PI_name PI, in Properties props);
6 :     Properties getProperties( in PI_name PI);
7 :     void activate();
8 :     void deactivate();
9 :       …
10 : };
11 : local interface RequestInfo {
12:      string target;
13:      string operation;
14 :    …
15 : };
16 : local interface ClientInterceptor : Interceptor {
17:      void sendRequest (in RequestInfo ri);
18:      void receiveReply (in RequestInfo ri);
19: };
20:
21: local interface ServerInterceptor : Interceptor {
22:      void receiveRequest  (in RequestInfo ri);
23       void sendReply (in RequestInfo ri);
24 };
```

Figure 7. The excerpt of portable interceptor interface definition (OMG IDL format).

The definition of **ClientInterceptor** is shown in Line 16-19 in Figure 7. The *ClientInterceptor*::*sendRequest*(*RequestInfo*) is invoked by SOAP engine before the client request is sent to server. The interceptor programmers can implement their desired intercepting code in this function. Similarly, the *ClientInterceptor*::*receiveReply*(*RequestInfo*) is invoked by SOAP engine after the SOAP engine receiving response from server. The interceptor has chance to inspect the response.

The proposed **ServerInterceptor** is defined in Line 21-24 of Figure 7. This interface define the two server side interception points *ServerInterceptor*::*receiveRequest*() and *ServerInterceptor*::*sendReply*(). The behavior of this interface is similar to **ClientInterceptor**.

4. The Implementation of Portable Interceptor Mechanism

As discussed in previous section, our portable interceptor mechanism in SOAP has two parts: a SOAP engine featured portable interceptor management API and portable interceptor. We adopt two open source SOAP engine, Apache Axis 1.3[1] and Codehaus XFire [3], to implement our portable interceptor mechanism design. We will first present the implementation of our portable interceptor management first and then the proposed portable interceptor Java class for Apache Axis and Codehaus XFire.

The Figure 8 depicts the excerpt of interface definition of the portable interceptor in Java. We assume our portable interceptor mechanism is part of SOAP standard. We can define our portable interceptor mechanism as Package **org.w3c.PortableInterceptor**. This interface is directly converted from the IDL definition of PI_Manager. A possible implementation of PI_Manager is shown in Figure 9. The Line 7 and 8 show how to load and manage the given portable interceptor in *PI_Manager*::*plugIn*().

```
package org.w3c.PortableInterceptor;
public interface PI_Manager extends java.rmi.Remote{
   void plugIn(PI_name PI, PI_module pim);
   void setProperties(PI_name PI, Properties props);
   Properties getProperties(PI_name PI);
  boolean activate(PI_name PI);
  boolean deactivate(PI_name PI);
}
```

Figure 8. The excerpt of PI_Manager interface definition.

```
1 : package org.apache.axis.PIM;
2 : ...
3 : public java PI_ManagerImpl implements PI_Manager {
4 :    private Hashtable plugedInterceptor = new HashTable ;
5 :    void plugIn(PI_name PI, PI_module pim) {
6 :          …
7 :          ServerInterceptor si=Class.forName(pim);
8 :          plugedInterceptor.put(PI, si);
9 :          ..
10:    };
11:   void setProperties(PI_name PI, Properties props) {
12:         …
13:    };
14:   Properties getProperties(PI_name PI) {
15:         ...
16:    };
17:  boolean activate(PI_name PI) {
18:         ...
19:    };
20:  boolean deactivate(PI_name PI) {
21:         …
22:    };
23 :}
```

Figure 9. The code excerpt of PI_Manager implementation for Apache Axis .

Figure 10, Figure 11, and Figure 12 show the excerpt of interface definitions that are directly converted from the related IDL definition in Figure 7. A portable interceptor developer should implement these interfaces with their desired interception feature than they may to load and manage their portable interceptor by using PI_Manager.

```
package org.w3c.PortableInterceptor;
/* All potable interceptors implement this interface
public interface Interceptor
{
   void initialize() ;
   void destroy() ;
    void activate() ;
   void deactivate() ;
    ...
} // interface Interceptor
```

Figure 10. The excerpt of Interceptor interface definition in Java.

```
package org.w3c.PortableInterceptor;
public interface ClientInterceptor extends Interceptor
{
   void sendRequest(...) ;
   void receiveReply(...) ;
} // interface ClientInterceptor
```

Figure 11. The excerpt of ClientInterceptor interface definition in Java.

```
package org.w3c.PortableInterceptor;
public interface ServerInterceptor extends Interceptor
{
   void receiveRequest(...) ;
   void sendReply(...) ;
} // interface ServerInterceptor
```

Figure 12. The excerpt of ServerInterceptor interface definition in Java.

5. Portability Test and Performance Evaluation

There are two parts of evaluation for this work: portability test on Apache Axis and XFire; and the performance evaluation of our PIM. We use an interceptor implementation example based on the PIM to examine whether this interceptor can be installed on both Apache Axis and XFire or not. We design a series of performance experiments to evaluate the performance of our PIM. The results show the overhead of our PIM is insignificant to the SOAP engine.

Figure 14 depicts the experiment environment for testing portability of PIM. We implemented a dummy portable interceptor based on the server side portable interceptor interface **org.w3c.PortableInterceptor.ServerInterceptor** and store in Host H_1 and Host H_2. The code excerpt of the dummy portable interceptor is shown in Figure 13. The dummy portable interceptor will dump the SOAP message (Line 6 in Figure 13) when the interception function receiveRequest() is called. The Host H_1 and H_2 are installed the modified open source SOAP engine Axis and XFire respectively, as shown in the Figure. An administration AP *admAP* and a client are installed on Host H_0. The admAP invokes the plugin() and activate() to plug in and activate our test portable interceptor **PI** (the Step 1 to 4 in the figure). The client then invokes an echoString() request to the web services on Host H_1 and H_2. The dumped SOAP messages on Axis and XFire engine console show that the portable server interceptor is functioning on both SOAP engine. That is, our portable interceptor is portable to these SOAP engine.

0-7695-2685-3/06 $20.00 © 2006 IEEE

```
1:  package tw.edu.ntou.cs.dms.PortableInterceptor;
2 :  import javax.xml.soap.SOAPMessage;
3 :  import org.w3c.PortableInterceptor.ServerInterceptor;
4 :  public class ServerPI01 implements ServerInterceptor {
5 :  public void receiveRequest(SOAPMessage reqMsg) {
6 :          System.out.println(reqMsg.toString());
7 :  }
...
     }
}
```

Figure 13. The code excerpt of the dummy service portable interceptor example.

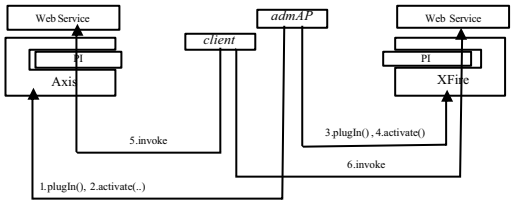


Figure 14. The environment to test the portability of PIM

Our PIM contributes runtime overhead when the PIM looks up the PI chain and call the interception functions in the interceptor. In order to examine the overheads of PIM, we use a dummy PI to examine the overhead on Axis and XFire respectively. The experiment results show the overheads on both Axis and XFire are all insignificant to the SOAP system.

The two set of experiments are conducts in the same experiment environment as shown in Figure 15. The PIM are implemented in both Apache Axis 1.3 and XFire 1.0 SOAP engine on a Pentium IV 3.2Ghz with 1GB memory PC. The host is running on MS Windows XP platform. The client is implemented in MS C# and invokes echoString() to the Web Services on the same host. The purpose of the experiment is to measure the response time delay doe the working PIM. Note that all results reported in this section have a 95% confidence interval with interval half-widths of less than 3% of average measurements.

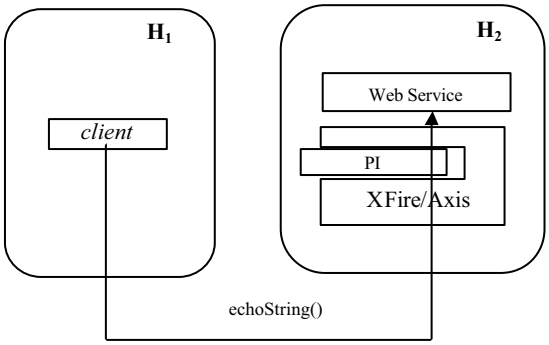


Figure 15. The environment of overhead experiment.

Based on the overhead analysis, we notice that the size of the input/output parameter in each nested invocation (or the message size) could affect the round trip time (RTT) of an invocation. Therefore, we measure the net processing time occurred in RAD for message size varying from 4 bytes to 500 KB. The experiment result, shown in Table 2, shows the client round trip time is increasing as the message size. We conclude that the overhead on Axis and XFire are all negligible.

Data Size	4 Bytes	1 KB	2 KB	5 KB	50 KB	500 KB
RTT with PIM (ms)	1.64	1.94	2.18	3.08	14.53	116.90
RTT w/o PIM (ms)	1.62	1.89	2.13	3.01	14.37	116.61
PIM overhead	0.02	0.04	0.05	0.07	0.15	0.29

(A) Axis

Data Size	4 Bytes	1 KB	2 KB	5 KB	50 KB	500 KB
RTT with PIM (ms)	1.10	1.11	1.19	1.53	6.18	98.55
RTT w/o PIM (ms)	1.09	1.11	1.19	1.52	6.18	99.01
PIM overhead	0.01	0.00	0.00	0.01	0.00	-0.46

(B) XFire

Table 2. The experiment data of client RTT delay.

6. Conclusions

In this paper, we have concluded the limitations of current SOAP standard in our previous fault tolerant Web Services research works and other works need interception mechanism as well. We take the advantages of SOAP to propose a Portable Interception Mechanism on SOAP. We also discussed the impacts on interception design due to the architectural differences in SOAP and CORBA. Our new PIM make SOAP interceptor portable like CORBA does. We believe that our experience on developing PIM can be applied to applications, such as security, etc., especially the communities familiar

0-7695-2685-3/06 $20.00

with other middleware.

References

1. Apache, Axis architecture Guide, Apache Axis 1.3 Documents, 2005.
2. Brown, K. and Kindel, C., Distributed Component Object Model Protocol – DCOM/1.0, 1996 http://www.microsoft.com/oledev/olecom/draft-brown-dcom-v1-spec-01.txt .
3. Codehaus, XFire user's guide, XFire 1.0 Document, 2006.
4. Fang, C.L., Liang, D., Chen, C., and Lin, P., A Redundant Nested Invocation Suppression Mechanism for Active Replication Fault Tolerant Web Service, in Proc. Of the IEEE International Conference on e-Technology, e-Commerce and e-Service (EEE' 04), pp. 9-16, Taipei, Taiwan, March 2004.
5. Anand Krishnan, Morgan Deters, Venkita Subramonian, Interceptor. cs562, Jan. 2002.
6. Liang, D., Fang, C., Chen, C., and Lin, F., 2003, Fault tolerant web service, 10th Asia-Pacific Software Engineering Conference (APSEC'03)
7. Liang, D., Fang, C., Yuan, S., Chen, C., A fault-tolerant object service on CORBA, The Journal of System and Software, vol. 48, pp.197-211, Nov, 1999.
8. Liang, D., Fang, C., Chen, C., Lin F., Fault tolerant web service, in Proc. of IEEE APSEC, 2003, pp. 310-321, Chiang Mai, Thailand, December 2003.
9. Lin, F., Liang, D., and Wu, S., A Study on Interceptor in Supporting Continuous Monitoring, to appear in Proc. Of 2006 Annual Meeting of the American Accounting Association, August, 2006.
10. Microsoft Inc., DCOM Technical documents, http://msdn.microsoft.com/library/, 2003.
11. Microsoft, ASP.NET Web Services or .NET Remoting: How to Choose, 2002. http://msdn.microsoft.com/library/default.asp?url=/ library/en-us/dnbda/html/bdadotnetarch16.asp.
12. Microsoft, Logging Application Block Introduction, http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnpag/html/logging-ch01.asp, 2003.
13. Object Management Group, Common Object Request Broker Architecture: Core Specification, V 3.0.3, 2004. OMG Technical Committee Document formal/04-03- 01.
14. Object Management Group, WSDL-SOAP to CORBA Interworking, OMG Technical Committee Document mars/03-05-07, 2003.
15. Salamone, S., Secure E-Markets Emerge: Extranet VPNs Mean EBusiness, Internet Week, May 8 2000 issue, 2000.
16. Song, D., dSniff Document, dSniff 2.3, 2001. http://www.monkey.org/~dugsong/dsniff/
17. Sun Microsystems, Java RMI Specification, 2003. http://java.sun.com/j2se/1.4.2/docs/guide/rmi/spec/rmiTOC.html
18. Vasarhelyi, M., and Halper, F., The continuous audit of online systems. Auditing: A Journal of Practice and Theory, Vol. 10, No. 1, pp.110-125, 1991.
19. W3C, Simple Object Access Protocol (SOAP) 1.1, 2000, http://www.w3.org/TR/SOAP/.
20. W3C, Web Services Description Language (WSDL) 1.1, 2001. http://www.w3.org/TR/wsdl
21. W3C, Simple Object Access Protocol (SOAP) 1.2 recommendation, 2003. http://www.w3.org/TR/2003/PR-soap12-part0-20030624/
22. Zwass, V., Electric Commerce: Structures and Issues, International Journal of Electric Commerce, pp. 3-23, 1996.

0-7695-2685-3/06 $20.00 © 2006 IEEE

Journal of Network and Computer Applications 34 (2011) 1–11

Contents lists available at ScienceDirect

Journal of Network and Computer Applications

journal homepage: www.elsevier.com/locate/jnca

Review

A survey on security issues in service delivery models of cloud computing

S. Subashini *, V. Kavitha

Anna University Tirunelveli, Tirunelveli, TN 627007, India

ARTICLE INFO

Article history:
Received 3 March 2010
Received in revised form
11 July 2010
Accepted 11 July 2010

Keywords:
Cloud computing
Data privacy
Data protection
Security
Virtualization

ABSTRACT

Cloud computing is a way to increase the capacity or add capabilities dynamically without investing in new infrastructure, training new personnel, or licensing new software. It extends Information Technology's (IT) existing capabilities. In the last few years, cloud computing has grown from being a promising business concept to one of the fast growing segments of the IT industry. But as more and more information on individuals and companies are placed in the cloud, concerns are beginning to grow about just how safe an environment it is. Despite of all the hype surrounding the cloud, enterprise customers are still reluctant to deploy their business in the cloud. Security is one of the major issues which reduces the growth of cloud computing and complications with data privacy and data protection continue to plague the market. The advent of an advanced model should not negotiate with the required functionalities and capabilities present in the current model. A new model targeting at improving features of an existing model must not risk or threaten other important features of the current model. The architecture of cloud poses such a threat to the security of the existing technologies when deployed in a cloud environment. Cloud service users need to be vigilant in understanding the risks of data breaches in this new environment. In this paper, a survey of the different security risks that pose a threat to the cloud is presented. This paper is a survey more specific to the different security issues that has emanated due to the nature of the service delivery models of a cloud computing system.

Contents

* Corresponding author. Tel.: +91 9840638819.
E-mail addresses: subasundararajan@gmail.com (S. Subashini), kavinayav@gmail.com (V. Kavitha).

1084-8045/$ - see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.jnca.2010.07.006

1. Introduction

Today Small and Medium Business (SMB) companies are increasingly realizing that simply by tapping into the cloud they can gain fast access to best business applications or drastically boost their infrastructure resources, all at negligible cost. Gartner (Jay Heiser, 2009) defines cloud computing (Stanojevi et al., 2008; Vaquero et al., 2009; Weiss, 2007; Whyman, 2008; Boss et al., 2009) as "a style of computing where massively scalable IT-enabled capabilities are delivered 'as a service' to external customers using Internet technologies". Cloud providers currently enjoy a profound opportunity in the marketplace. The providers must ensure that they get the security aspects right, for they are the ones who will shoulder the responsibility if things go wrong. The cloud offers several benefits like fast deployment, pay-for-use, lower costs, scalability, rapid provisioning, rapid elasticity, ubiquitous network access, greater resiliency, hypervisor protection against network attacks, low-cost disaster recovery and data storage solutions, on-demand security controls, real time detection of system tampering and rapid re-constitution of services. While the cloud offers these advantages, until some of the risks are better understood, many of the major players will be tempted to hold back (Viega, 2009). According to a recent IDCI survey, 74% of IT executives and CIO's cited security as the top challenge preventing their adoption of the cloud services model (Clavister, 2009). Analysts' estimate that within the next five years, the global market for cloud computing will grow to $95 billion and that 12% of the worldwide software market will move to the cloud in that period. To realize this tremendous potential, business must address the privacy questions raised by this new computing model (BNA, 2009). Cloud computing moves the application software and databases to the large data centers, where the management of the data and services are not trustworthy. This unique attribute, however, poses many new security challenges (Cong Wang et al., 2009). These challenges include but not limited to accessibility vulnerabilities, virtualization vulnerabilities, web application vulnerabilities such as SQL (Structured Query Language) injection and cross-site scripting, physical access issues, privacy and control issues arising from third parties having physical control of data, issues related to identity and credential management, issues related to data verification, tampering, integrity, confidentiality, data loss and theft, issues related to authentication of the respondent device or devices and IP spoofing.

Though cloud computing is targeted to provide better utilization of resources using virtualization techniques and to take up much of the work load from the client, it is fraught with security risks (Seccombe et al., 2009). The complexity of security risks in a complete cloud environment is illustrated in Fig. 1.

In Fig. 1, the lower layer represents the different deployment models of the cloud namely private, community, public and hybrid cloud deployment models. The layer just above the deployment layer represents the different delivery models that are utilized within a particular deployment model. These delivery models are the SaaS (Software as a Service), PaaS (Platform as a Service) and IaaS (Infrastructure as a Service) delivery models. These delivery models form the core of the cloud and they exhibit certain characteristics like on-demand self-service, multi-tenancy, ubiquitous network, measured service and rapid elasticity which are shown in the top layer. These fundamental elements of the cloud require security which depends and varies with respect to the deployment model that is used, the way by which it is delivered and the character it exhibits. Some of the fundamental security challenges are data storage security, data transmission

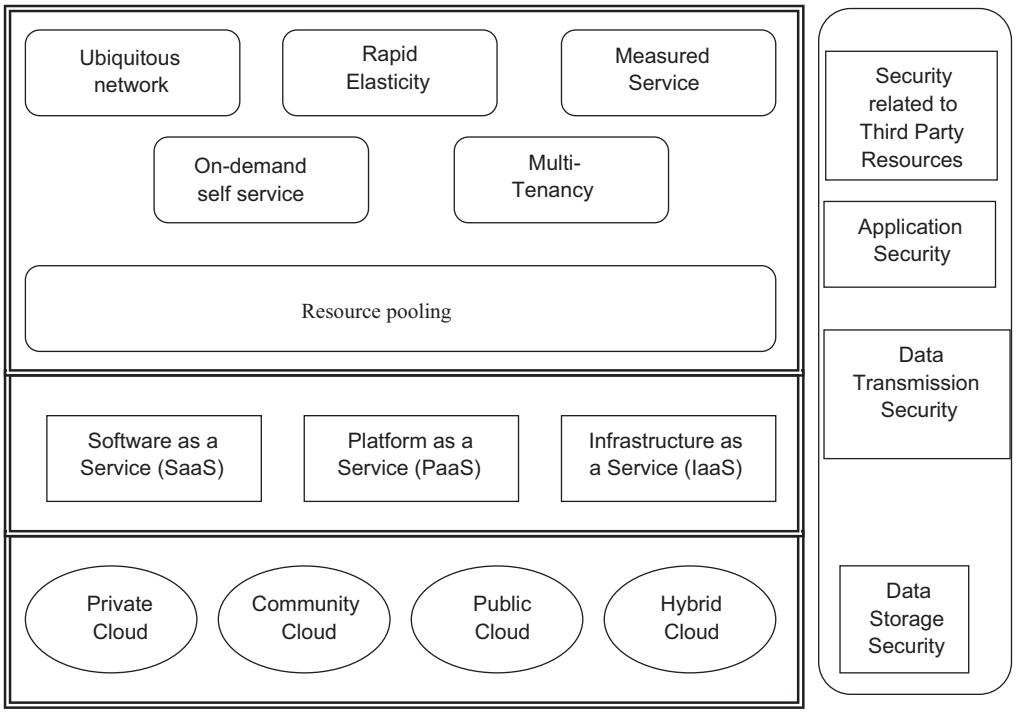


Fig. 1. Complexity of security in cloud environment.

security, application security and security related to third-party resources.

This paper is concentrated towards the issues related to the service delivery models. SaaS is a model of software deployment whereby a provider licenses an application to customers for use as a service on demand. One example of SaaS is the Salesforce.com CRM application. IaaS is the delivery of computer infrastructure (typically a platform virtualization environment) as a service. Rather than purchasing servers, software, data center space or network equipment, clients instead buy those resources as a fully outsourced service. One such example of this is the Amazon web services. PaaS is the delivery of a computing platform and solution stack as a service. It facilitates the deployment of applications without the cost and complexity of buying and managing the underlying hardware and software layers. PaaS provides the facilities required to support the complete lifecycle of building and delivering web applications and services. An example of this would be GoogleApps.

This paper describes the various security issues of cloud computing due to its service delivery models. In the first place, the underlying technology of cloud by itself provides a major security risk. This paper is organized as follows: Section 2 describes the common security issues that are posed by the cloud service delivery models. Section 3 describes the security threats posed by the "Software as a Service" (SaaS) delivery model. Section 4 describes the security threats posed by the "Platform as a Service" (PaaS) delivery model. Section 5 describes the security threats posed by the "Infrastructure as a Service" (IaaS) delivery model. Section 6 lists some of the current solutions which partly target the security challenges posed by the cloud. Section 7 provides conclusions derived out of this survey.

2. Security issues in service models

Cloud computing utilizes three delivery models by which different types of services are delivered to the end user. The three delivery models are the SaaS, PaaS and IaaS which provide infrastructure resources, application platform and software as services to the consumer. These service models also place a different level of security requirement in the cloud environment. IaaS is the foundation of all cloud services, with PaaS built upon it and SaaS in turn built upon it. Just as capabilities are inherited, so are the information security issues and risks. There are significant trade-offs to each model in the terms of integrated features, complexity vs. extensibility and security. If the cloud service provider takes care of only the security at the lower part of the security architecture, the consumers become more responsible for implementing and managing the security capabilities.

A recent survey by Cloud Security Alliance (CSA) & IEEE indicates that enterprises across sectors are eager to adopt cloud computing but that security are needed both to accelerate cloud adoption on a wide scale and to respond to regulatory drivers. It also details that cloud computing is shaping the future of IT but the absence of a compliance environment is having dramatic impact on cloud computing's growth. Organizations using cloud computing as a service infrastructure, critically like to examine the security and confidentiality issues for their business critical insensitive applications. Yet, guaranteeing the security of corporate data in the "cloud" is difficult, if not impossible, as they provide different services like SaaS, PaaS, and IaaS. Each service has its own security issues (Kandukuri et al., 2009).

SaaS is a software deployment model where applications are remotely hosted by the application or service provider and made available to customers on demand, over the Internet. The SaaS model offers the customers with significant benefits, such as improved operational efficiency and reduced costs. SaaS is rapidly emerging as the dominant delivery model for meeting the needs of enterprise IT services. However, most enterprises are still uncomfortable with the SaaS model due to lack of visibility about the way their data is stored and secured. According to the Forrester study, "The State of Enterprise Software: 2009," security concerns are the most commonly cited reason why enterprises are not interested in SaaS. Consequently, addressing enterprise security concerns has emerged as the biggest challenge for the adoption of SaaS applications in the cloud (Heidi Lo et al., 2009). However, to overcome the customer concerns about application and data security, vendors must address these issues head-on. There is a strong apprehension about insider breaches, along with vulnerabilities in the applications and systems' availability that could lead to loss of sensitive data and money. Such challenges can dissuade enterprises from adopting SaaS applications within the cloud.

IaaS completely changes the way developers deploy their applications. Instead of spending big with their own data centers or managed hosting companies or colocation services and then hiring operations staff to get it going, they can just go to Amazon Web Services or one of the other IaaS providers, get a virtual server running in minutes and pay only for the resources they use. With cloud brokers like Rightscale, enStratus, etc., they could easily grow big without worrying about things like scaling and additional security. In short, IaaS and other associated services have enabled startups and other businesses focus on their core competencies without worrying much about the provisioning and management of infrastructure. IaaS completely abstracted the hardware beneath it and allowed users to consume infrastructure as a service without bothering anything about the underlying complexities. The cloud has a compelling value proposition in terms of cost, but "out of the box" IaaS only provides basic security (perimeter firewall, load balancing, etc.) and applications moving into the cloud will need higher levels of security provided at the host.

PaaS is one layer above IaaS on the stack and abstracts away everything up to OS, middleware, etc. This offers an integrated set of developer environment that a developer can tap to build their applications without having any clue about what is going on underneath the service. It offers developers a service that provides a complete software development lifecycle management, from planning to design to building applications to deployment to testing to maintenance. Everything else is abstracted away from the "view" of the developers. The dark side of PaaS is that, these advantages itself can be helpful for a hacker to leverage the PaaS cloud infrastructure for malware command and control and go behind IaaS applications.

3. Security issues in SaaS

In SaaS, the client has to depend on the provider for proper security measures. The provider must do the work to keep multiple users' from seeing each other's data. So it becomes difficult to the user to ensure that right security measures are in place and also difficult to get assurance that the application will be available when needed (Choudhary, 2007). With SaaS, the cloud customer will by definition be substituting new software applications for old ones. Therefore, the focus is not upon portability of applications, but on preserving or enhancing the security functionality provided by the legacy application and achieving a successful data migration (Seccombe et al., 2009).

The SaaS software vendor may host the application on its own private server farm or deploy it on a cloud computing infrastructure service provided by a third-party provider (e.g. Amazon,

Google, etc.). The use of cloud computing coupled with the pay-as-you-go (grow) approach helps the application service provider reduce the investment in infrastructure services and enables it to concentrate on providing better services to customers.

Over the past decade, computers have become widespread within enterprises, while IT services and computing has become a commodity. Enterprises today view data and business processes (transactions, records, pricing information, etc.) themselves as strategic and guard them with access control and compliance policies. However, in the SaaS model, enterprise data is stored at the SaaS provider's data center, along with the data of other enterprises. Moreover, if the SaaS provider is leveraging a public cloud computing service, the enterprise data might be stored along with the data of other unrelated SaaS applications. The cloud provider might, additionally, replicate the data at multiple locations across countries for the purposes of maintaining high availability. Most enterprises are familiar with the traditional on-premise model, where the data continues to reside within the enterprise boundary, subject to their policies. Consequently, there is a great deal of discomfort with the lack of control and knowledge of how their data is stored and secured in the SaaS model. There are strong concerns about data breaches, application vulnerabilities and availability that can lead to financial and legal liabilities.

The layered stack for a typical SaaS vendor and critical aspects that must be covered across layers in order to ensure security of the enterprise data is illustrated in Fig. 2.

The following key security elements should be carefully considered as an integral part of the SaaS application development and deployment process:

- Data security
- Network security
- Data locality
- Data integrity
- Data segregation
- Data access
- Authentication and authorization

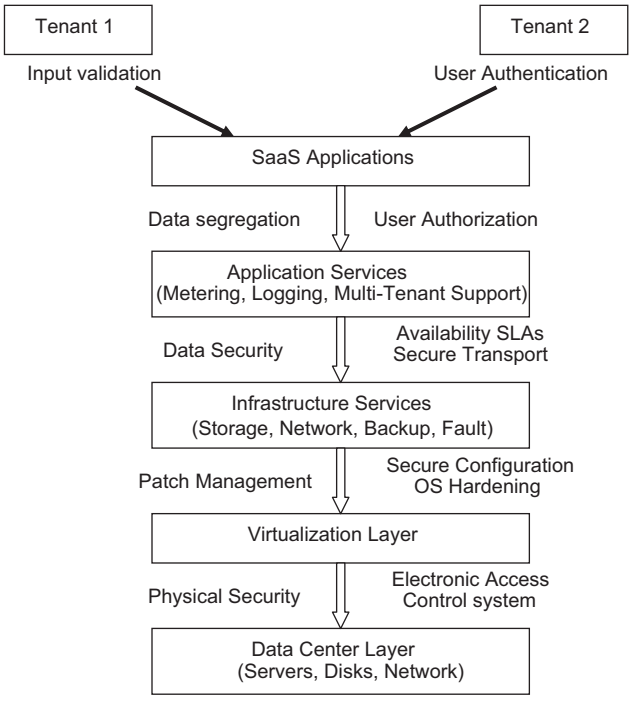


Fig. 2. Security for the SaaS stack.

- Data confidentiality
- Web application security
- Data breaches
- Virtualization vulnerability
- Availability
- Backup
- Identity management and sign-on process.

The different security issues of SaaS are discussed as follows.

3.1. Data security

In a traditional on-premise application deployment model, the sensitive data of each enterprise continues to reside within the enterprise boundary and is subject to its physical, logical and personnel security and access control policies. However, in the SaaS model, the enterprise data is stored outside the enterprise boundary, at the SaaS vendor end. Consequently, the SaaS vendor must adopt additional security checks to ensure data security and prevent breaches due to security vulnerabilities in the application or through malicious employees. This involves the use of strong encryption techniques for data security and fine-grained authorization to control access to data.

In cloud vendors such as Amazon, the Elastic Compute Cloud (EC2) administrators do not have access to customer instances and cannot log into the Guest OS. EC2 Administrators with a business need are required to use their individual cryptographically strong Secure Shell (SSH) keys to gain access to a host. All such accesses are logged and routinely audited. While the data at rest in Simple Storage Service (S3) is not encrypted by default, users can encrypt their data before it is uploaded to Amazon S3, so that it is not accessed or tampered with by any unauthorized party.

Malicious users can exploit weaknesses in the data security model to gain unauthorized access to data. The following assessments test and validate the security of the enterprise data stored at the SaaS vendor:

- Cross-site scripting [XSS]
- Access control weaknesses
- OS and SQL injection flaws
- Cross-site request forgery [CSRF]
- Cookie manipulation
- Hidden field manipulation
- Insecure storage
- Insecure configuration.

Any vulnerability detected during these tests can be exploited to gain access to sensitive enterprise data and lead to a financial loss.

3.2. Network security

In a SaaS deployment model, sensitive data is obtained from the enterprises, processed by the SaaS application and stored at the SaaS vendor end. All data flow over the network needs to be secured in order to prevent leakage of sensitive information. This involves the use of strong network traffic encryption techniques such as Secure Socket Layer (SSL) and the Transport Layer Security (TLS) for security.

In case of Amazon WebServices (AWS), the network layer provides significant protection against traditional network security issues, such as MITM (Man-In-The-Middle) attacks, IP spoofing, port scanning, packet sniffing, etc. For maximum security, Amazon S3 is accessible via SSL encrypted endpoints. The

encrypted endpoints are accessible from both the Internet and from within Amazon EC2, ensuring that data is transferred securely both within AWS and to and from sources outside of AWS.

However, malicious users can exploit weaknesses in network security configuration to sniff network packets. The following assessments test and validate the network security of the SaaS vendor:

- Network penetration and packet analysis
- Session management weaknesses
- Insecure SSL trust configuration.

Any vulnerability detected during these tests can be exploited to hijack active sessions, gain access to user credentials and sensitive data.

3.3. Data locality

In a SaaS model of a cloud environment, the consumers use the applications provided by the SaaS and process their business data. But in this scenario, the customer does not know where the data is getting stored. In many a cases, this can be an issue. Due to compliance and data privacy laws in various countries, locality of data is of utmost importance in many enterprise architecture (Softlayer, 2009). For example, in many EU and South America countries, certain types of data cannot leave the country because of potentially sensitive information. In addition to the issue of local laws, there's also the question of whose jurisdiction the data falls under, when an investigation occurs. A secure SaaS model must be capable of providing reliability to the customer on the location of the data of the consumer.

3.4. Data integrity

Data integrity is one of the most critical elements in any system. Data integrity is easily achieved in a standalone system with a single database. Data integrity in such a system is maintained via database constraints and transactions. Transactions should follow ACID (atomicity, consistency, isolation and durability) properties to ensure data integrity. Most databases support ACID transactions and can preserve data integrity.

Next in the complexity chain are distributed systems. In a distributed system, there are multiple databases and multiple applications. In order to maintain data integrity in a distributed system, transactions across multiple data sources need to be handled correctly in a fail safe manner. This can be done using a central global transaction manger. Each application in the distributed system should be able to participate in the global transaction via a resource manager. This can be achieved using a 2-phase commit protocol as per XA standard.

Enter the world of SOA and Cloud computing, and the problem of the data integrity gets magnified even more, as there is a mix of on-premise and SaaS applications exposed as service. SaaS applications are multi-tenant applications hosted by a third party. SaaS applications usually expose their functionality via XML based APIs (Application Program Interfaces). Also, in SOA based environments, many on-premise applications expose their functionality via SOAP and REST web services as well. One of the biggest challenges with web services is transaction management. At the protocol level, HTTP (Hyper Text Transfer Protocol) does not support transactions or guaranteed delivery, so the only option is to implement these at the API level. Although there are standards available for managing data integrity with web services such as WS-Transaction and WS-Reliability, these standards are not yet mature and not many vendors have implemented these. Most SaaS vendors expose their web services APIs without any support for transactions. Also, each SaaS application may have different levels of availability and SLA (service-level agreement), which further complicates management of transactions and data integrity across multiple SaaS applications.

The lack of integrity controls at the data level (or, in the case of existing integrity controls, bypassing the application logic to access the database directly) could result in profound problems. Architects and developers need to approach this danger cautiously, making sure they do not compromise databases' integrity in their zeal to move to cloud computing.

3.5. Data segregation

Multi-tenancy is one of the major characteristics of cloud computing. As a result of multi-tenancy multiple users can store their data using the applications provided by SaaS. In such a situation, data of various users will reside at the same location. Intrusion of data of one user by another becomes possible in this environment. This intrusion can be done either by hacking through the loop holes in the application or by injecting client code into the SaaS system. A client can write a masked code and inject into the application. If the application executes this code without verification, then there is a high potential of intrusion into other's data. A SaaS model should therefore ensure a clear boundary for each user's data. The boundary must be ensured not only at the physical level but also at the application level. The service should be intelligent enough to segregate the data from different users.

A malicious user can use application vulnerabilities to handcraft parameters that bypass security checks and access sensitive data of other tenants. The following assessments test and validate the data segregation of the SaaS vendor in a multi-tenant deployment:

- SQL injection flaws
- Data validation
- Insecure storage.

Any vulnerability detected during these tests can be exploited to gain access to sensitive enterprise data of other tenants.

3.6. Data access

Data access issue is mainly related to security policies provided to the users while accessing the data. In a typical scenario, a small business organization can use a cloud provided by some other provider for carrying out its business processes. This organization will have its own security policies based on which each employee can have access to a particular set of data. The security policies may entitle some considerations wherein some of the employees are not given access to certain amount of data. These security policies must be adhered by the cloud to avoid intrusion of data by unauthorized users (Blaze et al., 1999; Kormann and Rubin, 2000; Bowers et al., 2008). The SaaS model must be flexible enough to incorporate the specific policies put forward by the organization. The model must also be able to provide organizational boundary within the cloud because multiple organization will be deploying their business processes within a single cloud environment.

3.7. Authentication and authorization

Most companies, if not all, are storing their employee information in some type of Lightweight Directory Access Protocol (LDAP) servers. In the case of SMB companies, a segment that has the highest SaaS adoption rate, Active Directory (AD) seems to be the most popular tool for managing users (Microsoft White Paper, 2010). With SaaS, the software is hosted outside of the corporate firewall. Many a times user credentials are stored in the SaaS providers' databases and not as part of the corporate IT infrastructure. This means SaaS customers must remember to remove/disable accounts as employees leave the company and create/enable accounts as come onboard. In essence, having multiple SaaS products will increase IT management overhead. For example, SaaS providers can provide delegate the authentication process to the customer's internal LDAP/AD server, so that companies can retain control over the management of users.

3.8. Data confidentiality issue

The definitional borders of cloud computing are much debated today. Cloud computing involves the sharing or storage by users of their own information on remote servers owned or operated by others and accesses through the Internet or other connections. Cloud computing services exist in many variations, including data storage sites, video sites, tax preparation sites, personal health record websites and many more. The entire contents of a user's storage device may be stored with a single cloud provider or with many cloud providers. Whenever an individual, a business, a government agency, or any other entity shares information in the cloud, privacy or confidentiality questions arise. Some of the findings related to the confidentiality issues are:

1. Cloud computing has significant implications for the privacy of personal information as well as for the confidentiality of business and governmental information.
2. A user's privacy and confidentiality risks vary significantly with the terms of service and privacy policy established by the cloud provider.
3. For some types of information and some categories of cloud computing users, privacy and confidentiality rights, obligations, and status may change when a user discloses information to a cloud provider.
4. Disclosure and remote storage may have adverse consequences for the legal status of protections for personal or business information.
5. The location of information in the cloud may have significant effects on the privacy and confidentiality protections of information and on the privacy obligations of those who process or store the information.
6. Information in the cloud may have more than one legal location at the same time with differing legal consequences.
7. Laws could oblige a cloud provider to examine user records for evidence of criminal activity and other matters.
8. Legal uncertainties make it difficult to assess the status of information in the cloud as well as the privacy and confidentiality protections available to users.

In an electronic environment, the Electronic Communications Privacy Act of 1986 (ECPA) provides some protections against government access to electronic mail and other computer records held by third parties. The privacy protections available under ECPA for the wide range of cloud computing activities are difficult to predict. Indeed, simply identifying all cloud computing applications would be a significant challenge by itself. Factors that may affect the proper applications of ECPA to cloud computing activities include

1. The precise characterization of the activity as a communication or as a storage, complicated by the recognition that an activity can move from being a communication to being store communication depending on time and possibly other factors.
2. Whether the information in question is content or non-content (e.g., header or transaction information).
3. The terms of service established by the cloud provider.
4. Any consent that the user has granted to the provider or others.
5. The identity of the service provider, for example, if the cloud provider is itself a government agency, the provider's obligation would be different from those of a non-governmental cloud provider, and the rights of users would be different.

3.9. Web application security

SaaS is software deployed over the internet and/or is deployed to run behind a firewall in local area network or personal computer. The key characteristics include Network-based access to, and management of, commercially available software and managing activities from central locations rather than at each customer's site, enabling customers to access application remotely via the Web. SaaS application development may use various types of software components and frameworks. These tools can reduce time-to-market and the cost of converting a traditional on-premise software product or building and deploying a new SaaS solution. Examples include components for subscription management, grid computing software, web application frameworks and complete SaaS platform products. One of the "must-have" requirements for a SaaS application is that it has to be used and managed over the web (in a browser) (Michal Zalewski, 2009). The software which is provided as a service resides in the cloud without tying up with the actual users. This allows improvising the software without inconveniencing the user. Security holes in the web applications thus create a vulnerability to the SaaS application. In this scenario, the vulnerability can potentially have detrimental impact on all of the customers using the cloud. The challenge with SaaS security is not any different than with any other web application technology, however one of the problems is that traditional network security solutions such as network firewalls, network intrusion detection and prevention systems (IDS & IPS), do not adequately address the problem. Web applications introduce new security risks that cannot effectively be defended against at the network level, and do require application level defenses.

Verizon Business in their 'Verizon Business 2008 Data Breach Investigation Report' (Wade et al., 2008) reported 59% of the breaches involve hacking with the following breakdown:

- Application/service layer—39%
- OS/platform layer—23%
- Exploit known vulnerability—18%
- Exploit unknown vulnerability—5%
- Use of back door—5%.

Attacks targeting applications, software, and services were by far the most common techniques, representing 39% of all hacking activity leading to data compromise. This follows a trend in recent years of attacks moving up the stack. Far from past, operating system, platform, and server-level attacks accounted for a sizable portion of breaches. Eighteen percent of hacks exploited a specific

known vulnerability while 5% exploited unknown vulnerabilities for which a patch was not available at the time of the attack. Evidence of re-entry via backdoors, which enable prolonged access and control of compromised systems, was found in 15% of hacking-related breaches. The attractiveness of this to criminals desiring large quantities of information is obvious.

SQL injection (Robert Auger, 2009) is one type of attack which makes the web application more vulnerable. If the application is vulnerable to such type of attacks, the entire data behind the application is at risk. The data can be either belonging to the organization from where the attack is launched or it can also be private data of some other organization hosted in the same cloud.

Since the web applications and SaaS are tightly coupled in providing services to the cloud users, most of the security threats of web application are also posed by the SaaS model of the cloud. The Open Web Application Security Project has identified Top 10 security risks faced by web applications. Those threats are:

1. Injection flaws like SQL, OS and LDAP injection
2. Cross-site scripting
3. Broken authentication and session management
4. Insecure direct object references
5. Cross-site request forgery
6. Security misconfiguration
7. Insecure cryptographic storage
8. Failure to restrict URL access
9. Insufficient transport layer protection
10. Unvalidated redirects and forwards.

3.10. Data breaches

Since data from various users and business organizations lie together in a cloud environment, breaching into the cloud environment will potentially attack the data of all the users. Thus the cloud becomes a high value target (Bernard Golden, 2009; Kaufman, 2009). In the Verizon Business breach report blog (Russ Cooper, 2008) it has been stated that external criminals pose the greatest threat (73%), but achieve the least impact (30,000 compromised records), resulting in a Psuedo Risk Score of 67,500. Insiders pose the least threat (18%), and achieve the greatest impact (375,000 compromised records), resulting in a Pseudo Risk Score of 67,500. Partners are middle in both (73.39% and 187,500) resulting in a Pseudo Risk Score of 73,125. Though SaaS advocates claim that SaaS providers can provide better security to customers' data than by conventional means, Insiders still have access to the data but it is just that they are accessing it in a different way. Insiders do not have direct access to databases, but it does not reduce the risk of insider breaches which can be a massive impact on the security. The SaaS providers' employees have access to a lot more information and a single incident could expose information from many customers. SaaS providers must be compliant with PCI DSS (Payment Card Industry—Data Security Standards) (PCI DSS, 2009) in order to host merchants that must comply with PCI DSS.

3.11. Vulnerability in virtualization

Virtualization is one of the main components of a cloud. But this poses major security risks. Ensuring that different instances running on the same physical machine are isolated from each other is a major task of virtualization which is not met completely in today's scenario. The other issue is the control of administrator on host and guest operating systems. Current VMMs (Virtual Machine Monitor) do not offer perfect isolation. Many bugs have been found in all popular VMMs that allow escaping from VM. Virtual machine monitor should be 'root secure', meaning that no privilege within the virtualized guest environment permits interference with the host system.

Some vulnerability has been found in all virtualization software which can be exploited by malicious, local users to bypass certain security restrictions or gain privileges. For example, the vulnerability of Microsoft Virtual PC and Microsoft Virtual Server could allow a guest operating system user to run code on the host or another guest operating system. Vulnerability in Virtual PC and Virtual Server could allow elevation of privilege. Another example would be the vulnerability in Xen caused due to an input validation error in tools/pygrub/src/GrubConf.py. This can be exploited by 'root' users of a guest domain to execute arbitrary commands in domain 0 via specially crafted entries in grub.conf when the guest system is booted. A perfection of properties like isolation, inspection and interposition is yet to be completely achieved in VMMs.

3.12. Availability

The SaaS application needs to ensure that enterprises are provided with service around the clock. This involves making architectural changes at the application and infrastructural levels to add scalability and high availability. A multi-tier architecture needs to be adopted, supported by a load-balanced farm of application instances, running on a variable number of servers. Resiliency to hardware/software failures, as well as to denial of service attacks, needs to be built from the ground up within the application.

At the same time, an appropriate action plan for business continuity (BC) and disaster recovery (DR) needs to be considered for any unplanned emergencies. This is essential to ensure the safety of the enterprise data and minimal downtime for enterprises.

With Amazon for instance, the AWS API endpoints are hosted on the same Internet-scale, world-class infrastructure that supports the Amazon.com retail site. Standard Distributed Denial of Service (DDoS) mitigation techniques such as synchronous cookies and connection limiting are used. To further mitigate the effect of potential DDoS attacks, Amazon maintains internal bandwidth that exceeds its provider-supplied Internet bandwidth.

These assessments test and validate the availability of the SaaS vendor.

- Authentication weaknesses
- Session management weaknesses.

Many applications provide safeguards to automatically lock user accounts after successive incorrect credentials. However, incorrect configuration and implementation of such features can be used by malicious users to mount denial of service attacks.

3.13. Backup

The SaaS vendor needs to ensure that all sensitive enterprise data is regularly backed up to facilitate quick recovery in case of disasters. Also the use of strong encryption schemes to protect the backup data is recommended to prevent accidental leakage of sensitive information.

In the case of cloud vendors such as Amazon, the data at rest in S3 is not encrypted by default. The users need to separately encrypt their data and backups so that it cannot be accessed or tampered with by unauthorized parties.

The following assessments test and validate the security of the data backup and recovery services provided by the SaaS vendor:

- Insecure storage
- Insecure configuration.

Any vulnerability detected during these tests can be exploited to gain access to sensitive enterprise data stored in backups.

3.14. Identity management and sign-on process

Identity management (IdM) or ID management is a broad administrative area that deals with identifying individuals in a system (such as a country, a network or an organization) and controlling the access to the resources in that system by placing restrictions on the established identities. Identity management can involve three perspectives

1. *The pure identity paradigm*: Creation, management and deletion of identities without regard to access or entitlements.
2. *The user access (log-on) paradigm*: For example: a smart card and its associated data used by a customer to log on to a service or services (a traditional view).
3. *The service paradigm*: A system that delivers personalized role-based, online, on-demand, multimedia (content), presence-based services to users and their devices.

The SaaS vendor can support identity management and sign on services using any of the following models.

3.14.1. Independent IdM stack

The SaaS vendor provides the complete stack of identity management and sign on services. All information related to user accounts, passwords, etc. is completely maintained at the SaaS vendor end.

3.14.2. Credential synchronization

The SaaS vendor supports replication of user account information and credentials between enterprise and SaaS application. The user account information creation is done separately by each tenant within the enterprise boundary to comply with its regulatory needs. Relevant portions of user account information are replicated to the SaaS vendor to provide sign on and access control capabilities. The authentication happens at the SaaS vendor end using the replicated credentials.

3.14.3. Federated IdM

The entire user account information including credentials is managed and stored independently by each tenant. The user authentication occurs within the enterprise boundary. The identity of the user as well as certain user attributes are propagated on-demand to the SaaS vendor using federation to allow sign on and access control.

The security challenges for adopting these models and the relative advantages and disadvantages are listed in Table 1.

The following assessments test and validate the security of the identity management and sign-on process of the SaaS vendor:

- Authentication weakness analysis
- Insecure trust configuration.

Any vulnerability detected during these tests can be exploited to take over user accounts and compromise sensitive data.

4. Security issues in PaaS

In PaaS, the provider might give some control to the people to build applications on top of the platform. But any security below the application level such as host and network intrusion prevention will still be in the scope of the provider and the provider has to offer strong assurances that the data remains inaccessible between applications. PaaS is intended to enable developers to build their own applications on top of the platform. As a result it tends to be more extensible than SaaS, at the expense of customer-ready features. This tradeoff extends to security features and capabilities, where the built-in capabilities are less complete, but there is more flexibility to layer on additional security.

Applications sufficiently complex to leverage an Enterprise Service Bus (ESB) need to secure the ESB directly, leveraging a

Table 1
Security challenges in identity management [IdM] and sign-on process.

IdM and SSO model	Advantages	Disadvantages	Security challenges
Independent IdM stack	• Easy to implement • No separate integration with enterprise directory	• The users need to remember separate credentials for each SaaS application	• The IdM stack should be highly configurable to facilitate compliance with enterprise policies; e.g., password strength, etc.
Credential synchronization	• Users do not need to remember multiple passwords	• Requires integration with enterprise directory • Has higher security risk value due to transmissions of user credentials outside enterprise perimeter	• The SaaS vendor needs to ensure security of the credentials during transit and storage and prevent their leakage
Federated IdM	• Users do not need to remember multiple passwords • No separate integration with enterprise directory • Low security risk value as compared to credential synch	• Relatively more complex to implement	• The SaaS vendor and tenants need to ensure that proper trust relationships and validations are established to ensure secure federation of user identities

protocol such as Web Service (WS) Security (Oracle, 2009). The ability to segment ESBs is not available in PaaS environments. Metrics should be in place to assess the effectiveness of the application security programs. Among the direct application, security specific metrics available are vulnerability scores and patch coverage. These metrics can indicate the quality of application coding. Attention should be paid to how malicious actors react to new cloud application architectures that obscure application components from their scrutiny. Hackers are likely to attack visible code, including but not limited to code running in user context. They are likely to attack the infrastructure and perform extensive black box testing. The vulnerabilities of cloud are not only associated with the web applications but also vulnerabilities associated with the machine-to-machine Service-Oriented Architecture (SOA) applications, which are increasingly being deployed in the cloud.

5. Security issues in IaaS

With IaaS the developer has better control over the security as long as there is no security hole in the virtualization manager. Also, though in theory virtual machines might be able to address these issues but in practice there are plenty of security problems (Attanasio, 1973; Gajek et al., 2007). The other factor is the reliability of the data that is stored within the provider's hardware. Due to the growing virtualization of 'everything' in information society, retaining the ultimate control over data to the owner of data regardless of its physical location will become a topic of utmost interest. To achieve maximum trust and security on a cloud resource, several techniques would have to be applied (Descher et al., 2009).

The security responsibilities of both the provider and the consumer greatly differ between cloud service models. Amazon's Elastic Compute Cloud (EC2) (Amazon, 2010) infrastructure as a service offering, as an example, includes vendor responsibility for security up to the hypervisor, meaning they can only address security controls such as physical security, environmental security, and virtualization security. The consumer, in turn, is responsible for the security controls that relate to the IT system including the OS, applications and data (Seccombe et al., 2009).

5.1. Impact of deployment model

IaaS is prone to various degrees of security issues based on the cloud deployment model through which it is being delivered. Public cloud poses the major risk whereas private cloud seems to have lesser impact. Physical security of infrastructure and disaster management if any damage is incurred to the infrastructure (either naturally or intentionally), is of utmost importance. Infrastructure not only pertains to the hardware where data is processed and stored but also the path where it is getting transmitted. In a typical cloud environment, data will be transmitted from source to destination through umpteen number of third-party infrastructure devices (Ristenpart et al., 2009). There is a high possibility that data can be routed through an intruder's infrastructure. The complexity involved in IaaS due to each of the service deployment models is illustrated in Table 2.

Although cloud architecture is an improvised technology, the underlying technologies remain the same. The cloud is just built over the internet and all the concerns related to security in internet are also posed by the cloud. The basis of the cloud technology makes the consumer and provider reside at different location and virtually access the resources over the Internet. Even if enormous amount of security is put in place in the cloud, still the data is transmitted through the normal underlying Internet technology. So, the security concerns which are threatening the Internet also threaten the cloud. But, in a cloud, the risks are overwhelmingly high. This is because of its vulnerability and the asset value of the resources and their nature of them residing together. Cloud systems still uses normal protocols and security measures that are used in the Internet but the requirements are at a higher extent. Encryption and secure protocols cater to the needs to a certain extent but they are not context oriented. A robust set of policies and protocols are required to help secure transmission of data within the cloud. Concerns regarding intrusion of data by external non users of the cloud through the internet should also be considered. Measures should be set in place to make the cloud environment secure, private and isolated in the Internet to avoid cyber criminals attacking the cloud.

6. Current security solutions

There are several research works happening in the area of cloud security. Several groups and organization are interested in developing security solutions and standards for the cloud. The Cloud Security Alliance (CSA) is gathering solution providers, non-profits and individuals to enter into discussion about the current and future best practices for information assurance in the cloud ("Cloud Security Alliance (CSA)—security best practices for cloud computing," 2009 (Cloud Security Alliance, 2010a, 2010b)). The Cloud Standards web site is collecting and coordinating information about cloud-related standards under development by the groups. The Open Web Application Security Project (OWASP) maintains list of top vulnerabilities to cloud-based or SaaS models which is updated as the threat landscape changes ("OWASP", 2010). The Open Grid Forum publishes documents to containing security and infrastructural specifications and information for grid computing developers and researchers ("Open Grid Forum", 2010).

The best security solution for web applications is to develop a development framework that has tough security architecture. Tsai W, Jin Z, and Bai X, put forth a four-tier framework for web-based development that though seems interesting, only implies a security facet in the process (Tsai et al., 2009). "Towards best practices in designing for the cloud" by Berre, Roman, Landre, Heuval, Skar, Udnaes, Lennon, and Zeid is a road map toward cloud-centric development (Berre et al., 2009), and the X10 language is one way to achieve better use of cloud capabilities of

Table 2
Cloud service deployment model.

	Infrastructure management	Infratructure ownership	Infrastructure location	Access and consumption
Public cloud	Third-party provider	Third-party provider	Off-premise	Untrusted
Private/community cloud	Organization or third-party provider	Organization or third-party provider	On-premise or off-premise	Trusted
Hybrid cloud	Both organization and third-party provider	Both organization and third-party provider	Both on-premise and off-premise	Trusted and untrusted

massive parallel processing and concurrency (Saraswat Vijay, 2010).

Krugel et al. (2002) point out the value of filtering a packet-sniffer output to specific services as an effective way to address security issues shown by anomalous packets directed to specific ports or services (Krugel et al., 2002). An often-ignored solution to accessibility vulnerabilities is to shut down unused services, keep patches updated, and reduce permissions and access rights of applications and users (Krugel et al., 2002).

Raj et al. (2009) suggest resource isolation to ensure security of data during processing, by isolating the processor caches in virtual machines, and isolating those virtual caches from the hypervisor cache (Raj et al., 2009). Hayes points out that there is no way to know if the cloud providers properly deleted a client's purged data, or whether they saved it for some unknown reason (Hayes, 2008).

Basta and Halton (2007) suggest one way to avoid IP spoofing by using encrypted protocols wherever possible. They also suggest avoiding ARP poisoning by requiring root access to change ARP tables; using static, rather than dynamic ARP tables; or at least make sure changes to the ARP tables are logged.

Hayes (2008) points out an interesting wrinkle here, "Allowing a third-party service to take custody of personal documents raises awkward questions about control and ownership: If you move to a competing service provider, can you take a data with you? Could you lose access to documents if you fail to pay a bill?". The issues of privacy and control cannot be solved, but merely assured with tight service-level agreements (SLAs) or by keeping the cloud itself private.

One simple solution, which Milne (2010) states to be a widely used solution for UK businesses is to simply use in-house "private clouds" (Milne, 2010). Nurmi, Wolski, Grzegorczyk, Obertelli, Soman, Youseff, & Zagorodnov show a preview of one of the available home-grown clouds in their (2009) presentation "The Eucalyptus Open-Source Cloud-Computing System" (Nurmi et al., 2009).

7. Conclusion

As described in the paper, though there are extreme advantages in using a cloud-based system, there are yet many practical problems which have to be solved. Cloud computing is a disruptive technology with profound implications not only for Internet services but also for the IT sector as a whole. Still, several outstanding issues exist, particularly related to service-level agreements (SLA), security and privacy, and power efficiency. As described in the paper, currently security has lot of loose ends which scares away a lot of potential users. Until a proper security module is not in place, potential users will not be able to leverage the advantages of this technology. This security module should cater to all the issues arising from all directions of the cloud. Every element in the cloud should be analyzed at the macro and micro level and an integrated solution must be designed and deployed in the cloud to attract and enthrall the potential consumers. Until then, cloud environment will remain cloudy.

An integrated security model targeting different levels of security of data for a typical cloud infrastructure is under research. This model is meant to be more dynamic and localized in nature. My research questions will center on application and data security over the cloud, and I intend to develop a framework by which the security methodology varies dynamically from one transaction/communication to another. One of the pieces of the framework might be focused on providing data security by storing and accessing data based on meta-data information. This would be more like storing related data in different locations based on the meta-data information which would make information invaluable if a malicious intent user recovers it. Keeping this as a core concept I am doing research on a framework which would be practical. Another piece of the framework would be providing 'Security as a Service' to the applications by providing security as a single-tier or a multi-tier based on the application's requirement and addition to it, the tiers are enabled to change dynamically making the security system less predictable. This research is based on the conceptualization of the cloud security based on real world security system where in security depends on the requirement and asset value of an individual or organization. For example, a normal human does not require personal security but a well known personality needs a body guard, an organization needs a set of security persons and a state or country have their mass military to safe guard their assets. The intense of security is directly proportional to the value of the asset it guards. In a cloud where there are heterogeneous systems having a variation in their asset value, a single security system would be too costly for certain applications and if there is less security then the vulnerability factor of some applications like financial and military applications will shoot up. On the other side, if the cloud has a common security methodology in place, it will be a high value asset target for hackers because of the fact that hacking the security system will make the entire cloud vulnerable to attack. In such a scenario, if customized security is provided as a service to applications, it would make sense. Though there are many practical concerns regarding to dynamic security and data storage based on meta-data information my research is much concentrated to derive a framework which targets these concepts and provide a practical solution.

References

Amazon. Amazon Elastic Compute Cloud (EC2), 2010 〈http://www.amazon.com/ec2/〉 [accessed: 10 December 2009].

Attanasio CR. Virtual machines and data security. In: Proceedings of the workshop on virtual computer systems. New York, NY, USA: ACM; 1973. p. 206–9.

Auger R. SQL Injection, 2009 〈http://projects.webappsec.org/SQL-Injection〉 [accessed on: 15 February 2010].

Basta A, Halton W. Computer security and penetration testing. Delmar Cengage Learning 2007.

Bernard Golden. Defining private clouds, 2009 〈http://www.cio.com/article/492695/Defining_Private_Clouds_Part_One〉 [accessed on: 11 January 2010].

Berre AJ, Roman D, Landre E, Heuvel WVD, Skar LA, Udnaes M, et al. Towards best practices in designing for the cloud. In: Proceedings of the 24th ACM SIGPLAN conference companion on object oriented programming systems languages and applications, Orlando, Florida, USA, 2009, p. 697–8.

Blaze M, Feigenbaum J, loannidis J, Keromytis AD. The role of trust management in distributed systems security, secure Internet programming, issues for mobile and distributed objects. Berlin: Springer-Verlag; 1999. p. 185–210.

BNA. Privacy & security law report, 8 PVLR 10, 03/09/2009. Copyright 2009 by The Bureau of National Affairs, Inc. (800-372-1033), 2009 〈http://www.bna.com〉 [accessed on: 2 November 2009].

Boss G, Malladi P, Quan D, Legregni L, Hall H. Cloud computing, 2009, p. 4 〈http://www.ibm.com/developerswork/websphere/zones/ hipods/library.html〉 [accessed on: 18 October 2009].

Bowers KD, Juels A, Oprea A. HAIL: a high-availability and integrity layer for cloud storage, Cryptology ePrint Archive, Report 2008/489, 2008 〈http://eprint.iacr.org/〉 [accessed on: 18 October 2009].

Choudhary V. Software as a service: implications for investment in software development. In: International conference on system sciences, 2007, p. 209.

Clavister. Security in the cloud, Clavister White Paper 〈http://www.it-wire.nu/members/cla69/attachments/CLA_WP_SECURITY_IN_THE_CLOUD.pdf〉 [accessed on: 21 October 2009].

Cloud Security Alliance. Guidance for identity & access management V2.1, 2010a 〈http://www.cloudsecurityalliance.org/guidance/csaguide-dom12-v2.10.pdf〉 [accessed on: 9 May 2010].

Cloud Security Alliance. Security best practices for cloud computing, 2010b 〈http://www.cloudsecurityalliance.org〉 [accessed on: 10 April 2010].

Cooper R. Verizon Business Data Breach security blog, 2008 〈http://securityblog.verizonbusiness.com/2008/06/10/2008-data-breach-investigations-report/〉 [accessed on: 11 February 2010].

Descher M, Masser P, Feilhauer T, Tjoa AM, Huemer D. Retaining data control to the client in infrastructure clouds. In: International conference on availability, reliability and security, ARES '09, 2009, p. 9–16.

An Efficient and Secure Dynamic Auditing Protocol for Data Storage in Cloud Computing

Kan Yang, *Student Member, IEEE*, Xiaohua Jia, *Senior Member, IEEE*

Abstract—In cloud computing, data owners host their data on cloud servers and users (data consumers) can access the data from cloud servers. Due to the data outsourcing, however, this new paradigm of data hosting service also introduces new security challenges, which requires an independent auditing service to check the data integrity in the cloud. Some existing remote integrity checking methods can only serve for static archive data and thus cannot be applied to the auditing service since the data in the cloud can be dynamically updated. Thus, an efficient and secure dynamic auditing protocol is desired to convince data owners that the data are correctly stored in the cloud. In this paper, we first design an auditing framework for cloud storage systems and propose an efficient and privacy-preserving auditing protocol. Then, we extend our auditing protocol to support the data dynamic operations, which is efficient and provably secure in the random oracle model. We further extend our auditing protocol to support batch auditing for both multiple owners and multiple clouds, without using any trusted organizer. The analysis and simulation results show that our proposed auditing protocols are secure and efficient, especially it reduce the computation cost of the auditor.

Index Terms—Storage Auditing, Dynamic Auditing, Privacy-Preserving Auditing, Batch Auditing, Cloud Computing.

1 Introduction

Cloud storage is an important service of cloud computing [1], which allows data owners (owners) to move data from their local computing systems to the cloud. More and more owners start to store the data in the cloud [2]. However, this new paradigm of data hosting service also introduces new security challenges [3]. Owners would worry that the data could be lost in the cloud. This is because data loss could happen in any infrastructure, no matter what high degree of reliable measures cloud service providers would take [4]–[8]. Sometimes, cloud service providers might be dishonest. They could discard the data which has not been accessed or rarely accessed to save the storage space and claim that the data are still correctly stored in the cloud. Therefore, owners need to be convinced that the data are correctly stored in the cloud.

Traditionally, owners can check the data integrity based on two-party storage auditing protocols [9]–[17]. In cloud storage system, however, it is inappropriate to let either side of cloud service providers or owners conduct such auditing, because none of them could be guaranteed to provide unbiased auditing result. In this situation, *third party auditing* is a natural choice for the storage auditing in cloud computing. A third party auditor (auditor) that has expertise and capabilities can do a more efficient work and convince both cloud service providers and owners.

For the third party auditing in cloud storage systems, there are several important requirements which have been proposed in some previous works [18], [19]. The auditing protocol should have the following properties: 1) *Confidentiality*. The auditing protocol should keep owner's data confidential against the auditor. 2) *Dynamic Auditing*. The auditing protocol should support the dynamic updates of the data in the cloud. 3) *Batch Auditing*. The auditing protocol should also be able to support the batch auditing for multiple owners and multiple clouds.

Recently, several remote integrity checking protocols were proposed to allow the auditor to check the data integrity on the remote server [20]–[28]. Table 1 gives the comparisons among some existing remote integrity checking schemes in terms of the performance, the privacy protection, the support of dynamic operations and the batch auditing for multiple owners and multiple clouds. From Table 1, we can find that many of them are not privacy-preserving or cannot support the data dynamic operations, so that they cannot be applied to cloud storage systems.

In [23], the authors proposed a dynamic auditing protocol that can support the dynamic operations of the data on the cloud servers, but this method may leak the data content to the auditor because it requires the server to send the linear combinations of data blocks to the auditor. In [24], the authors extended their dynamic auditing scheme to be privacy-preserving and support the batch auditing for multiple owners. However, due to the large number of data tags, their auditing protocols may incur a heavy storage overhead on the server. In [25], Zhu *et al.* proposed a cooperative provable data possession scheme that can support the batch auditing for multiple clouds and also extend it to support the dynamic auditing in [26]. However, their scheme cannot support the batch auditing for multiple owners. That is because parameters for generating the data tags used by each owner are different and thus they cannot combine the data tags from multiple owners to conduct the batch

• *Kan Yang and Xiaohua Jia are with the Department of Computer Science, City University of Hong Kong, Kowloon, Hong Kong Email: kanyang3@student.cityu.edu.hk, csjia@cityu.edu.hk*

Digital Object Indentifier 10.1109/TPDS.2012.278

TABLE 1
Comparison of Remote Integrity Checking Schemes

Scheme	Computation		Commu-nication	Privacy	Dynamic	Batch Operation		Prob. of Detection
	Sever	Verifier				multi-owner	multi-cloud	
PDP [20]	$O(t)$	$O(t)$	$O(1)$	Yes	No	No	No	$1-(1-\rho)^t$
CPDP [21]	$O(t+s)$	$O(t+s)$	$O(t+s)$	No	No	No	No	$1-(1-\rho)^{ts}$
DPDP [22]	$O(t\log n)$	$O(t\log n)$	$O(t\log n)$	No	No	No	No	$1-(1-\rho)^t$
Audit [23], [24]	$O(t\log n)$	$O(t\log n)$	$O(t\log n)$	Yes	Yes	Yes	No	$1-(1-\rho)^t$
IPDP [25], [26]	$O(ts)$	$O(t+s)$	$O(t+s)$	Yes	Yes	No	Yes	$1-(1-\rho)^{ts}$
Our Scheme	$O(ts)$	$O(t)$	$O(t)$	Yes	Yes	Yes	Yes	$1-(1-\rho)^{ts}$

n is the total number of data blocks of a file; t is the number of challenged data blocks in an auditing query; s is the number of sectors in each data block; ρ is the probability of block/sector corruption (suppose the probability of corruption is the same for the equal size of data block or sector)

auditing. Another drawback is that their scheme requires an additional trusted organizer to send a commitment to the auditor during the multi-cloud batch auditing, because their scheme applies the mask technique to ensure the data privacy. However, such additional organizer is not practical in cloud storage systems. Furthermore, both Wang's schemes and Zhu's schemes incur heavy computation cost of the auditor, which makes the auditor a performance bottleneck.

In this paper, we propose an efficient and secure dynamic auditing protocol, which can meet the above listed requirements. To solve the data privacy problem, our method is to generate an encrypted proof with the challenge stamp by using the Bilinearity property of the bilinear pairing, such that the auditor cannot decrypt it but can verify the correctness of the proof. Without using the mask technique, our method does not require any trusted organizer during the batch auditing for multiple clouds. On the other hand, in our method, we let the server compute the proof as an intermediate value of the verification, such that the auditor can directly use this intermediate value to verify the correctness of the proof. Therefore, our method can greatly reduce the computing loads of the auditor by moving it to the cloud server.

Our original contributions can be summarized as follows.

1) We design an auditing framework for cloud storage systems and propose a privacy-preserving and efficient storage auditing protocol. Our auditing protocol ensures the data privacy by using cryptography method and the Bilinearity property of the bilinear pairing, instead of using the mask technique. Our auditing protocol incurs less communication cost between the auditor and the server. It also reduces the computing loads of the auditor by moving it to the server.
2) We extend our auditing protocol to support the data dynamic operations, which is efficient and provably secure in the random oracle model.
3) We further extend our auditing protocol to support batch auditing for not only multiple clouds but also multiple owners. Our multi-cloud batch auditing does not require any additional trusted organizer. The multi-owner batch auditing can greatly improve the auditing performance, especially in large scale cloud storage systems.

The remaining of this paper is organized as follows. In Section 2, we describe definitions of the system model and security model. In Section 3, we propose an efficient and inherently secure auditing protocol and extend it to support the dynamic auditing in Section 4. We further extend our auditing protocol to support the batch auditing for multiple owners and multiple clouds in Section 5. Section 6 give the performance analysis of our proposed auditing protocols in terms of communication cost and computation cost. The security proof will be shown in the supplemental file. In Section 7, we give the related work on storage auditing. Finally, the conclusion is given in Section 8.

2 Preliminaries and Definitions

In this section, we first describe the system model and give the definition of storage auditing protocol. Then, we define the threat model and security model for storage auditing system.

2.1 Definition of System Model

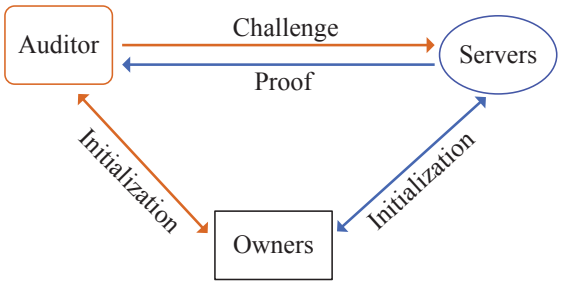


Fig. 1. System Model of the Data Storage Auditing

We consider an auditing system for cloud storage as shown in Fig.1, which involves data owners (owner), the

cloud server (server) and the third party auditor (auditor). The owners create the data and host their data in the cloud. The cloud server stores the owners' data and provides the data access to users (data consumers). The auditor is a trusted third party that has expertise and capabilities to provide data storage auditing service for both the owners and servers. The auditor can be a trusted organization managed by the government, which can provide unbiased auditing result for both data owners and cloud servers.

Before describing the auditing protocol definition, we first define some notations as listed in Table 2.

TABLE 2
Notations

Symbol	Physical Meaning
sk_t	secret tag key
pk_t	public tag key
sk_h	secret hash key
M	data component
T	set of data tags
n	number of blocks in each component
s	number of sectors in each data block
M_{info}	abstract information of M
$\mathcal{C}$	challenge generated by the auditor
$\mathcal{P}$	proof generated by the server

Definition 1 (Storage Auditing Protocol)**.** *A storage auditing protocol consists of the following five algorithms:* KeyGen, TagGen, Chall, Prove *and* Verify.

KeyGen$(\lambda) \rightarrow (sk_h, sk_t, pk_t)$. *This key generation algorithm takes no input other than the implicit security parameter* λ. *It outputs a secret hash key* sk_h *and a pair of secret-public tag key* (sk_t, pk_t).

TagGen$(M, sk_t, sk_h) \rightarrow T$. *The tag generation algorithm takes as inputs an encrypted file M, the secret tag key* sk_t *and the secret hash key* sk_h. *For each data block* m_i, *it computes a data tag* t_i *based on* sk_h *and* sk_t. *It outputs a set of data tags* $T = \{t_i\}_{i\in[1,n]}$.

Chall$(M_{info}) \rightarrow \mathcal{C}$. *The challenge algorithm takes as input the abstract information of the data* M_{info} *(e.g., file identity, total number of blocks, version number and timestamp etc.). It outputs a challenge* $\mathcal{C}$.

Prove$(M, T, \mathcal{C}) \rightarrow \mathcal{P}$. *The prove algorithm takes as inputs the file M, the tags T and the challenge from the auditor* $\mathcal{C}$. *It outputs a proof* $\mathcal{P}$.

Verify$(\mathcal{C}, \mathcal{P}, sk_h, pk_t, M_{info}) \rightarrow 0/1$. *The verification algorithm takes as inputs the* $\mathcal{P}$ *from the server, the secret hash key* sk_h, *the public tag key* pk_t *and the abstract information of the data* M_{info}. *It outputs the auditing result as* 0 *or* 1.

2.2 Definition of Security Model

We assume the auditor is honest-but-curious. It performs honestly during the whole auditing procedure but it is curious about the received data. But the sever could be dishonest and may launch the following attacks:

1) *Replace Attack*. The server may choose another valid and uncorrupted pair of data block and data tag (m_k, t_k) to replace the challenged pair of data block and data tag (m_i, t_i), when it already discarded m_i or t_i.
2) *Forge Attack*. The server may forge the data tag of data block and deceive the auditor, if the owner's secret tag keys are reused for the different versions of data.
3) *Replay Attack*. The server may generate the proof from the previous proof or other information, without retrieving the actual owner's data.

3 Efficient and Privacy-preserving Auditing Protocol

In this section, we first present some techniques we applied in the design of our efficient and privacy-preserving auditing protocol. Then, we describe the algorithms and the detailed construction of our auditing protocol for cloud storage systems. The correctness proof will be shown in the supplemental file.

3.1 Overview of Our Solution

The main challenge in the design of data storage auditing protocol is the *data privacy problem* (i.e., the auditing protocol should protect the data privacy against the auditor.). This is because: 1) For public data, the auditor may obtain the data information by recovering the data blocks from the data proof. 2) For encrypted data, the auditor may obtain content keys somehow through any special channels and could be able to decrypt the data. To solve the data privacy problem, our method is to generate an encrypted proof with the challenge stamp by using the Bilinearity property of the bilinear pairing, such that the auditor cannot decrypt it. But the auditor can verify the correctness of the proof without decrypting it.

Although the auditor has sufficient expertise and capabilities to conduct the auditing service, the computing ability of an auditor is not as strong as cloud servers. Since the auditor needs to audit for many cloud servers and a large number of data owners, the auditor could be the performance bottleneck. In our method, we let the server compute the proof as an intermediate value of the verification (calculated by the challenge stamp and the linear combinations of data blocks), such that the auditor can use this intermediate value to verify the proof. Therefore, our method can greatly reduce the computing loads of the auditor by moving it to the cloud server.

To improve the performance of an auditing system, we apply the *Data Fragment Technique* and *Homomorphic Verifiable Tags* in our method. The data fragment technique can reduce number of data tags, such that it can reduce the storage overhead and improve the system performance. By using the homomorphic verifiable tags, no matter how many data blocks are challenged, the server only responses the sum of data blocks and the product of tags to the auditor, whose size is constant and equal to only one data block. Thus, it reduces the communication cost.

3.2 Algorithms for Auditing Protocol

Suppose a file F has m data components as $F = (F_1, \cdots, F_m)$. Each data component has its physical meanings and can be updated dynamically by the data owners. For public data components, the data owner does not need to encrypted it, but for private data component, the data owner needs to encrypt it with its corresponding key.

Each data component F_k is divided into n_k data blocks denoted as $F_k = (m_{k1}, m_{k2}, \cdots, m_{kn_k})$. Due to the security reason, the data block size should is restricted by the security parameter. For example, suppose the security level is set to be 160-bit (20Byte), the data block size should be 20-Byte. A 50-KByte data component will be divided into 2500 data blocks and generate 2500 data tags, which incurs 50-KByte storage overhead.

By using the data fragment technique, we further split each data block into sectors. The sector size is restricted by the security parameter. We generate one data tag for each data block which consists of s sectors, such that less data tags are generated. In the same example above, a 50 KByte data component only incurs $50/s$ KByte storage overhead. In real storage systems, the data block size can be various. That is different data blocks could have different number of sectors. For example, if a data block m_i will be frequently read, then s_i could be large, but for those frequently updated data blocks, s_i could be relatively small.

For simplicity, we only consider one data component in our construction and constant number of sectors for each data block. Suppose there is a data component M, which is divided into n data blocks and each data block is further split into s sectors. For data blocks that have different number of sectors, we first select the maximum number of sectors s_{max} among all the sector numbers s_i. Then, for each data block m_i with s_i sectors, $s_i < s_{max}$, we simply consider that the data block m_i has s_{max} sectors by setting $m_{ij} = 0$ for $s_i < j \leq s_{max}$. Because the size of each sector is constant and equal to the security parameter p, we can calculate the number of data blocks as $n = \frac{sizeof(M)}{s \cdot \log p}$. We denote the encrypted data component as $M = \{m_{ij}\}_{i \in [1,n], j \in [1,s]}$.

Let $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ be the multiplicative groups with the same prime order p and $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ be the bilinear map. Let g_1 and g_2 be the generators of $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively. Let $h : \{0,1\}^* \to \mathbb{G}_1$ be a keyed secure hash function that maps the M_{info} to a point in $\mathbb{G}_1$.

Our storage auditing protocol consists of the following algorithms:

KeyGen$(\lambda) \to (pk_t, sk_t, sk_h)$. The key generation algorithm takes no input other than the implicit security parameter λ. It chooses two random number $sk_t, sk_h \in \mathbb{Z}_p$ as the secret tag key and the secret hash key. It outputs the public tag key as $pk_t = g_2^{sk_t} \in \mathbb{G}_2$, the secret tag key sk_t and the secret hash key sk_h.

TagGen$(M, sk_t, sk_h) \to T$. The tag generation algorithm takes each data component M, the secret tag key sk_t and the secret hash key sk_h as inputs. It first chooses s random values $x_1, x_2, \cdots, x_s \in \mathbb{Z}_p$ and computes $u_j = g_1^{x_j} \in \mathbb{G}_1$ for all $j \in [1,s]$. For each data block $m_i (i \in [1,n])$, it computes a data tag t_i as

$$t_i = (h(sk_h, W_i) \cdot \prod_{j=1}^{s} u_j^{m_{ij}})^{sk_t},$$

where $W_i = FID||i$ (the "||" denotes the concatenation operation), in which FID is the identifier of the data and i represents the block number of m_i. It outputs the set of data tags $T = \{t_i\}_{i \in [1,n]}$.

Chall$(M_{info}) \to \mathcal{C}$. The challenge algorithm takes the abstract information of the data M_{info} as the input. It selects some data blocks to construct the *Challenge Set* Q and generates a random number $v_i \in \mathbb{Z}_p^*$ for each chosen data block $m_i (i \in Q)$. Then, it computes the challenge stamp $R = (pk_t)^r$ by randomly choosing a number $r \in \mathbb{Z}_p^*$. It outputs the challenge as $\mathcal{C} = (\{i, v_i\}_{i \in Q}, R)$.

Prove$(M, T, \mathcal{C}) \to \mathcal{P}$. The prove algorithm takes as inputs the data M and the received challenge $\mathcal{C} = (\{i, v_i\}_{i \in Q}, R)$. The proof consists of the *tag proof* TP and the *data proof* DP. The *tag proof* is generated as

$$TP = \prod_{i \in Q} t_i^{v_i}.$$

To generate the *data proof*, it first computes the sector linear combination of all the challenged data blocks MP_j for each $j \in [1,s]$ as

$$MP_j = \sum_{i \in Q} v_i \cdot m_{ij}.$$

Then, it generates the *data proof* DP as

$$DP = \prod_{j=1}^{s} e(u_j, R)^{MP_j}.$$

It outputs the proof $\mathcal{P} = (TP, DP)$.

Verify$(\mathcal{C}, \mathcal{P}, sk_h, pk_t, M_{info}) \to 0/1$. The verification algorithm takes as inputs the challenge $\mathcal{C}$, the proof $\mathcal{P}$, the secret hash key sk_h, the public tag key pk_t and the abstract information of the data component. It first computes the identifier hash values $h(sk_h, W_i)$ of all the challenged data blocks and computes the challenge hash H_{chal} as

$$H_{chal} = \prod_{i \in Q} h(sk_h, W_i)^{rv_i}.$$

It then verifies the *proof* from the server by the following verification equation:

$$DP \cdot e(H_{chal}, pk_t) = e(TP, g_2^r). \quad (1)$$

If the above verification equation Eq.1 holds, it outputs 1. Otherwise, it outputs 0.

3.3 Construction of Our Privacy-preserving Auditing Protocol

As illustrated in Fig. 2, our storage auditing protocol consists of three phases: *Owner Initialization*, *Confirmation Auditing* and *Sampling Auditing*. During the system initialization, the owner generates the keys and the tags for the data. After storing the data on the server, the owner asks the auditor to conduct the confirmation auditing to make sure that their data is correctly stored on the server. Once

confirmed, the owner can choose to delete the local copy of the data. Then, the auditor conducts the sampling auditing periodically to check the data integrity.

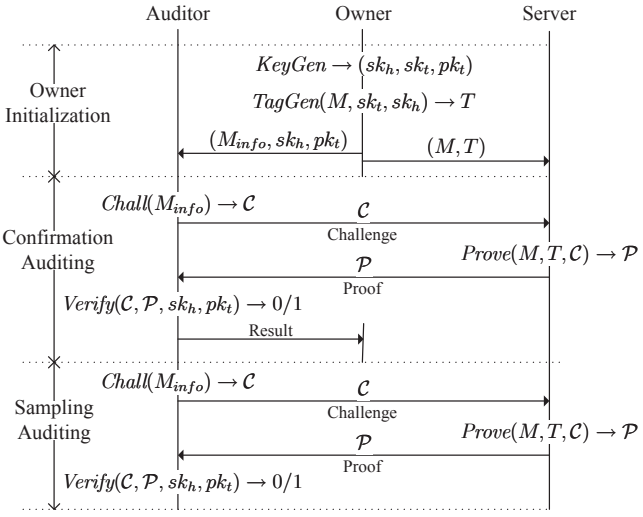


Fig. 2. Framework of Our Privacy-preserving Auditing Protocol

Phase 1: Owner Initialization

The owner runs the key generation algorithm KeyGen to generate the secret hash key sk_h, the pair of secret-public tag key (sk_t, pk_t). Then, it runs the tag generation algorithm TagGen to compute the data tags. After all the data tags are generated, the owner sends each data component $M = \{m_i\}_{i\in[1,n]}$ and its corresponding data tags $T = \{t_i\}_{i\in[1,n]}$ to the server together with the set of parameters $\{u_j\}_{j\in[1,s]}$. The owner then sends the public tag key pk_t, the secret hash key sk_h and the abstract information of the data M_{info} to the auditor, which includes the data identifier FID, the total number of data blocks n.

Phase 2: Confirmation Auditing

In our auditing construction, the auditing protocol only involves two-way communication: Challenge and Proof. During the confirmation auditing phase, the owner requires the auditor to check whether the owner's data is correctly stored on the server. The auditor conducts the confirmation auditing phase as

1) The auditor runs the challenge algorithm Chall to generate the challenge $\mathcal{C}$ for all the data blocks in the data component and sends the $\mathcal{C} = (\{i, v_i\}_{i\in Q}, R)$ to the server.
2) Upon receiving the challenge $\mathcal{C}$ from the auditor, the server runs the prove algorithm Prove to generate the proof $\mathcal{P} = (TP, DP)$ and sends it back to the auditor.
3) When the auditor receives the proof $\mathcal{P}$ from the server, it runs the verification algorithm Verify to check the correctness of $\mathcal{P}$ and extract the auditing result.

The auditor then sends the auditing result to the owner. If the result is true, the owner is convinced that its data is correctly stored on the server and it may choose to delete the local version of the data.

Phase 3: Sampling Auditing

The auditor will carry out the sampling auditing periodically by challenging a sample set of data blocks. The frequency of taking auditing operation depends on the service agreement between the data owner and the auditor (and also depends on how much trust the data owner has over the server). Similar to the confirmation auditing in Phase 2, the sampling auditing procedure also contains two-way communication as illustrated in Fig. 2.

Suppose each sector will be corrupted with a probability of ρ on the server. For a sampling auditing involved with t challenged data blocks, the probability of detection can be calculated as

$$Pr(t,s) = 1 - (1-\rho)^{t\cdot s}.$$

That is this t-block sampling auditing can detect any data corruption with a probability of $Pr(t,s)$.

4 Secure Dynamic Auditing

In cloud storage systems, the data owners will dynamically update their data. As an auditing service, the auditing protocol should be designed to support the dynamic data, as well as the static archive data. However, the dynamic operations may make the auditing protocols insecure. Specifically, the server may conduct two following attacks: 1) *Replay Attack.* The server may not update correctly the owner's data on the server and may use the previous version of the data to pass the auditing. 2) *Forge Attack.* When the data owner updates the data to the current version, the server may get enough information from the dynamic operations to forge the data tag. If the server could forge the data tag, it can use any data and its forged data tag to pass the auditing.

4.1 Our Solution

To prevent the replay attack, we introduce an *Index Table* (ITable) to record the abstract information of the data. The ITable consists of four components: *Index*, B_i, V_i and T_i. The *Index* denotes the current block number of data block m_i in the data component M. B_i denotes the original block number of data block m_i and V_i denotes the current version number of data block m_i. T_i is the timestamp used for generating the data tag.

This ITable is created by the owner during the owner initialization and managed by the auditor. When the owner completes the data dynamic operations, it sends an update message to the auditor for updating the ITable which is stored on the auditor. After the confirmation auditing, the auditor sends the result to the owner for the confirmation that the owner's data on the server and the abstraction information on the auditor are both up-to-date. This completes the data dynamic operation.

To deal with the forge attack, we can modify the tag generation algorithm TagGen. Specifically, when generating the data tag t_i for the data block m_i, we insert all the abstract information into the data tag by setting $W_i = FID||i||B_i||V_i||T_i$, such that the server cannot get enough information to forge the data tag from dynamic operations. The detailed proof will be given in the supplemental file. ?

TABLE 3
ITable of the Abstract Information of Data M

(a) Initial Abstract Information of M.

Index	B_i	V_i	T_i
1	1	1	T_1
2	2	1	T_2
3	3	1	T_3
$\vdots$	$\vdots$	$\vdots$	$\vdots$
n	n	1	T_n

(b) After modifying m_2, V_2 and T_2 are updated

Index	B_i	V_i	T_i
1	1	1	T_1
2	2	2	T_2^*
3	3	1	T_3
$\vdots$	$\vdots$	$\vdots$	$\vdots$
n	n	1	T_n

(c) After inserting before m_2, all items before m_2 move backward with the index increased by 1.

Index	B_i	V_i	T_i
1	1	1	T_1
2	$n+1$	1	T_{n+1}
3	2	1	T_2
$\vdots$	$\vdots$	$\vdots$	$\vdots$
$n+1$	n	1	T_n

(d) After deleting m_2, all items after m_2 move forward with the index decreased by 1.

Index	B_i	V_i	T_i
1	1	1	T_1
2	3	1	T_3
3	4	1	T_4
$\vdots$	$\vdots$	$\vdots$	$\vdots$
$n-1$	n	1	T_n

4.2 Algorithms and Constructions for Dynamic Auditing

The dynamic auditing protocol consists of four phases: Owner Initialization, Confirmation Auditing, Sampling Auditing and Dynamic Auditing.

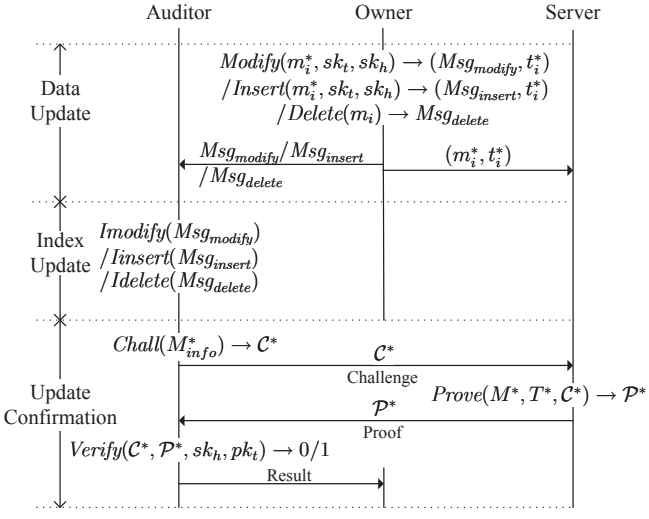


Fig. 3. Framework of Auditing for Dynamic Operations

The first three phases are similar to our privacy-preserving auditing protocol as described in the above section. The only differences are the tag generation algorithm TagGen and the ITable generation during the owner initialization phase. Here, as illustrated in Fig. 3, we only describe the dynamic auditing phase, which contains three steps: Data Update, Index Update and Update Confirmation.

Step 1: Data Update

There are three types of data update operations that can be used by the owner: Modification, Insertion and Deletion. For each update operation, there is a corresponding algorithm in the dynamic auditing to process the operation and facilitate the future auditing, defined as follows.

Modify$(m_i^*, sk_t, sk_h) \rightarrow (Msg_{modify}, t_i^*)$. The modification algorithm takes as inputs the new version of data block m_i^*, the secret tag key sk_t and the secret hash key sk_h. It generates a new version number V_i^*, new timestamp T_i^* and calls the TagGen to generate a new data tag t_i^* for data block m_i^*. It outputs the new tag t_i^* and the update message $Msg_{modify} = (i, B_i, V_i^*, T_i^*)$. Then, it sends the new pair of data block and tag (m_i^*, t_i^*) to the server and sends the update message Msg_{modify} to the auditor.

Insert$(m_i^*, sk_t, sk_h) \rightarrow (Msg_{insert}, t_i^*)$. The insertion algorithm takes as inputs the new data block m_i^*, the secret tag key sk_t and the secret hash key sk_h. It inserts a new data block m_i^* before the i-th position. It generates an original number B_i^*, a new version number V_i^* and a new timestamp T_i^*. Then, it calls the TagGen to generate a new data tag t_i^* for the new data block m_i^*. It outputs the new tag t_i^* and the update message $Msg_{insert} = (i, B_i^*, V_i^*, T_i^*)$. Then, it inserts the new pair of data block and tag (m_i^*, t_i^*) on the server and sends the update message Msg_{insert} to the auditor.

Delete$(m_i) \rightarrow Msg_{delete}$. The deletion algorithm takes as input the data block m_i. It outputs the update message $Msg_{delete} = (i, B_i, V_i, T_i)$. It then deletes the pair of data block and its tag (m_i, t_i) from the server and sends the update message Msg_{delete} to the auditor.

Step 2: Index Update

Upon receiving the three types of update messages, the auditor calls three corresponding algorithms to update the ITable. Each algorithm is designed as follows.

IModify(Msg_{modify}). The index modification algorithm takes the update message Msg_{modify} as input. It replaces the version number V_i by the new one V_i^* and modifies T_i by the new timestamp T_i^*.

IInsert(Msg_{insert}). The index insertion algorithm takes as input the update message Msg_{insert}. It inserts a new record (i, B_i^*, V_i^*, T_i^*) in i-th position in the ITable. It then moves the original i-th record and other records after the i-th position in the previous ITable backward in order, with the index number increased by one.

IDelete(Msg_{delete}). The index deletion algorithm takes as input the update message Msg_{delete}. It deletes the i-th record (i, B_i, V_i, T_i) in the ITable and all the records after the i-th position in the original ITable moved forward in order, with the index number decreased by one.

Table 3 shows the change of ITable according to the different type of data update operation. Table 3(a) describe the initial table of the data $M = \{m_1, m_2, \cdots, m_n\}$ and Table 3(b) describes the ITable after m_2 is updated. Table 3(c) is the ITable after a new data block is insert before m_2 and Table 3(d) shows the ITable after m_2 is deleted.

Step 3: Update Confirmation

After the auditor updates the ITable, it conducts a confirmation auditing for the updated data and sends the result to the owner. Then, the owner can choose to delete the local version of data according to the update confirmation auditing result.

5 Batch Auditing for Multi-owner and Multi-cloud

Data storage auditing is a significant service in cloud computing which helps the owners check the data integrity on the cloud servers. Due to the large number of data owners, the auditor may receive many auditing requests from multiple data owners. In this situation, it would greatly improve the system performance, if the auditor could combine these auditing requests together and only conduct the batch auditing for multiple owners simultaneously. The previous work [25] cannot support the batch auditing for multiple owners. That is because parameters for generating the data tags used by each owner are different and thus the auditor cannot combine the data tags from multiple owners to conduct the batch auditing.

On the other hand, some data owners may store their data on more than one cloud servers. To ensure the owner's data integrity in all the clouds, the auditor will send the auditing challenges to each cloud server which hosts the owner's data, and verify all the proofs from them. To reduce the computation cost of the auditor, it is desirable to combine all these responses together and do the batch verification.

In the previous work [25], the authors proposed a cooperative provable data possession for integrity verification in multi-cloud storage. In their method, the authors apply the mask technique to ensure the data privacy, such that it requires an additional trusted organizer to send a commitment to the auditor during the commitment phase in multi-cloud batch auditing. In our method, we apply the encryption method with the Bilinearity property of the bilinear pairing to ensure the data privacy, rather than the mask technique. Thus, our multi-cloud batch auditing protocol does not have any commitment phase, such that our method does not require any additional trusted organizer.

5.1 Algorithms for Batch Auditing for Multi-owner and multi-cloud

Let O be the set of owners and S be the set of cloud servers. The batch auditing for multi-owner and multi-cloud can be constructed as follows.

Phase 1: Owner Initialization

Each owner $O_k(k \in O)$ runs the key generation algorithm KeyGen to generate the pair of secret-public tag key $(sk_{t,k}, pk_{t,k})$ and a set of secret hash key $\{sk_{h,kl}\}_{l\in S}$. That is, for different cloud servers, the owner has different secret hash keys. We denote each data component as M_{kl}, which means that this data component is owned by the owner O_k and stored on the cloud server S_l. Suppose the data component M_{kl} is divided into n_{kl} data blocks and each data block is further split into s sectors. (Here we assume that each data block is further split into the same number of sectors. We can use the similar technique proposed in Section 3.2 to deal with the situation that each data blocks is split into different number of sectors.) The owner O_k runs the tag generation algorithm TagGen to generate the data tags $T_{kl} = \{t_{kl,i}\}_{i\in[1,n_{kl}]}$ as

$$t_{kl,i} = (h(sk_{h,kl}, W_{kl,i}) \cdot \prod_{j=1}^{s} u_{k,j}^{m_{kl,ij}})^{sk_{t,k}}.$$

where $W_{kl,i} = FID_{kl}||i||B_{kl,i}||V_{kl,i}||T_{kl,i}$.

After all the data tags are generated, each owner $O_k(k \in O)$ sends the data component $M_{kl} = \{m_{kl,ij}\}_{i\in[1,n_{kl}],j\in[1,s]}^{k\in O,l\in S}$ and the data tags $T_{kl} = \{t_{kl,i}\}_{i\in[1,n_{kl}]}^{k\in O,l\in S}$ to the corresponding server S_l. Then, it sends the public tag key $pk_{t,k}$, the set of secret hash key $\{sk_{hl,k}\}_{l\in S}$, the abstract information of data $\{M_{info,kl}\}_{k\in O,l\in S}$ to the auditor.

Phase 2: Batch Auditing for Multi-owner and Multi-Cloud

Let O_{chal} and S_{chal} denote the involved set of owners and cloud servers involved in the batch auditing respectively. The batch auditing also consists of three steps: Batch Challenge, Batch Proof and Batch Verification.

Step 1: Batch Challenge

During this step, the auditor runs the batch challenge algorithm BChall to generate a batch challenge $\mathcal{C}$ for a set of challenged owners O_{chal} and a set of clouds S_{chal}. The batch challenge algorithm is defined as follows.

BChall$(\{M_{info,kl}\}_{k\in O,l\in S}) \rightarrow \mathcal{C}$. The batch challenge algorithm takes all the abstract information as input. It selects a set of owners O_{chal} and a set of cloud servers S_{chal}. For each data owner $O_k(k \in O_{chal})$, it chooses a set of data blocks as the challenged subset Q_{kl} from each server $S_l(l \in S_{chal})$. It then generates a random number $v_{kl,i}$ for each chosen data block $m_{kl,i}(k \in O_{chal}, l \in S_{chal}, i \in Q_{kl})$. It also chooses a random number $r \in \mathbb{Z}_p^*$ and computes the set of challenge stamp $\{R_k\}_{k\in O_{chal}=pk_{t,k}^r}$. It outputs the challenge as

$$\mathcal{C} = (\{\mathcal{C}_l\}_{l\in S_{chal}}, \{R_k\}_{k\in O_{chal}}),$$

where $\mathcal{C}_l = \{(k, l, i, v_{kl,i})\}_{k\in O_{chal}}$.

Then, the auditor sends each $\mathcal{C}_l$ to each cloud server $S_l(l \in S_{chal})$ together with the challenge stamp $\{R_k\}_{k\in O_{chal}}$.

Step 2: Batch Proof

Upon receiving the challenge, each server $S_l(l \in S_{chal})$ generates a proof $\mathcal{P}_l = (TP_l, DP_l)$ by using the following batch prove algorithm BProve and sends the proof $\mathcal{P}_l$ to the auditor.

BProve$(\{M_{kl}\}_{k\in O_{chal}}, \{T_{kl}\}_{k\in O_{chal}}, \mathcal{C}_l, \{R_k\}_{k\in O_{chal}}) \rightarrow \mathcal{P}_l$. The batch prove algorithm takes as inputs the data $\{M_{kl}\}_{k\in O_{chal}}$, the data tags $\{T_{kl}\}_{k\in O_{chal}}$, the received challenge $\mathcal{C}_l$ and the challenge stamp $\{R_k\}_{k\in O_{chal}}$. It generates the *tag proof* TP_l as

$$TP_l = \prod_{k\in O_{chal}} \prod_{i\in Q_{kl}} t_{kl,i}^{v_{kl,i}}.$$

Then, for each $j \in [1, s]$, it computes the sector linear combination $MP_{kl,j}$ of all the chosen data blocks of each

owner $O_k(k \in O_{chal})$ as

$$MP_{kl,j} = \sum_{i \in Q_{kl}} v_{kl,i} \cdot m_{kl,ij},$$

and generates the *data proof* DP_l as

$$DP_l = \prod_{j=1}^{s} \prod_{k \in O_{chal}} e(u_{k,j}, R_k)^{MP_{kl,j}}.$$

It outputs the proof $\mathcal{P}_l = (TP_l, DP_l)$.

Step 3: Batch Verification

Upon receiving all the proofs from the challenged servers, the auditor runs the following batch verification algorithm BVerify to check the correctness of the proofs.

BVerify$(\mathcal{C}, \{\mathcal{P}_l\}, \{sk_{h,lk}\}, \{pk_{t,k}\}, \{M_{info,kl}\}) \rightarrow 0/1$. The batch verification algorithm takes as inputs the challenge $\mathcal{C}$, the proofs $\{\mathcal{P}_l\}_{l \in S_{chal}}$, the set of secret hash keys $\{sk_{h,kl}\}_{k \in O_{chal}, l \in S_{chal}}$, the public tag keys $\{pk_{t,k}\}_{k \in O_{chal}}$ and the abstract information of the challenged data blocks $\{M_{info,kl}\}_{k \in O_{chal}, l \in S_{chal}}$. For each owner $O_k(k \in O_{chal})$, it computes the set of identifier hash values $\{h(sk_{h,kl}, W_{kl,i})\}_{l \in S_{chal}, i \in Q_{kl}}$ for all the chosen data blocks from each challenged server, and use these hash values to compute a *challenge hash* $H_{chal,k}$ as

$$H_{chal,k} = \prod_{l \in S_{chal}} \prod_{i \in Q_{kl}} h(sk_{h,kl}, W_{kl,i})^{rv_{kl,i}}.$$

When finished the calculation of all the data owners' challenge hash $\{H_{chal,k}\}_{k \in O_{chal}}$, it verifies the proofs by the batch verification equation as

$$\prod_{l \in S_{chal}} DP_l = \frac{e(\prod_{l \in S_{chal}} TP_l, g_2^r)}{\prod_{k \in O_{chal}} e(H_{chal,k}, pk_{t,k})}. \quad (2)$$

If Eq.2 is true, it outputs 1. Otherwise, it outputs 0.

6 Performance Analysis of Our Auditing Protocols

Storage auditing is a very resource demanding service in terms of computational resource, communication cost and memory space. In this section, we give the communication cost comparison and computation complexity comparison between our scheme and two existing works: the Audit protocol proposed by Wang *et al.* [23], [24] and the IPDP proposed by Zhu *et al.* [25], [26]. The storage overhead analysis will be shown in the supplemental file.

6.1 Communication Cost

Because the communication cost during the initialization is almost the same in these three auditing protocols, we only compare the communication cost between the auditor and the server, which consists of the challenge and the proof.

Consider a batch auditing with K owners and C cloud servers. Suppose the number of challenged data block from each owner on different cloud servers is the same, denoted as t, and the data block are split into s sectors in Zhu's IPDP and our scheme. We do the comparison under the same probability of detection. That is, in Wang's scheme, the number of data blocks from each owner on each cloud server should be st. The result is described in Table 4.

TABLE 4
Communication Cost Comparison of Batch Auditing for K Owners and C Clouds

Scheme	Challenge	Proof
Wang's Audit [23], [24]	$O(KCst)$	$O(KCst \log n)$
Zhu's IPDP [25], [26]	$O(KCt)$	$O(KCs)$
Our Scheme	$O(KCt)$	$O(C)$

t is the number of challenged data blocks from each owner on each cloud server; s is the number of sectors in each data block; n is the total number of data blocks of a file in Wang's scheme.

From the table, we can see that the communication cost in Wang's auditing scheme is not only linear to C, K , t, s, but also linear to the total number of data blocks n. As we know, in large scale cloud storage systems, the total number of data blocks could be very large. Therefore, Wang's auditing scheme may incur high communication cost.

Our scheme and Zhu's IPDP have the same total communication cost during the challenge phase. During the proof phase, the communication cost of the proof in our scheme is only linear to C, but in Zhu's IPDP, the communication cost of the proof is not only linear to C and K, but also linear to s. That is because Zhu's IPDP uses the mask technique to protect the data privacy, which requires to send both the masked proof and the encrypted mask to the auditor. In our scheme, the server is only required to send the encrypted proof to the auditor and thus incurs less communication cost than Zhu's IPDP.

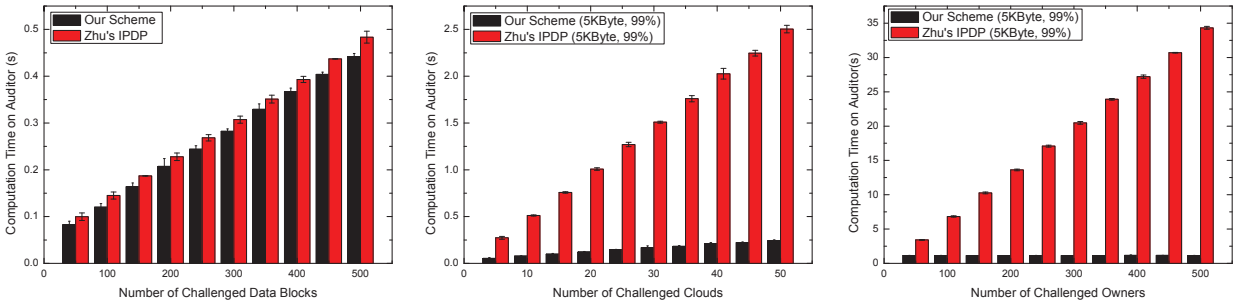
6.2 Computation Complexity

We simulate the computation of the owner, the server and the auditor on a Linux system with an Intel Core 2 Duo CPU at 3.16GHz and 4.00GB RAM. The code uses the Pairing-Based Cryptography (PBC) library version 0.5.12 to simulate our auditing scheme and Zhu's IPDP scheme (Under the same detection of probability, Wang's scheme requires much more data blocks than our scheme and Zhu's scheme, such that the computation time is almost s times more than our scheme and Zhu's IPDP and thus it is not comparable). The elliptic curve we used is a MNT $d159$-curve, where the base field size is 159-bit and the embedding degree is 6. The $d159$-curve has a 160-bit group order, which means p is a 160-bit length prime. All the simulation results are the mean of 20 trials.

6.2.1 Computation Cost of the Auditor

We compare the computation time of the auditor versus the number of data blocks, the number of clouds and the number of owners in Fig. 4.

Fig. 4(a) shows the computation time of the auditor versus the number of challenged data blocks in the single cloud and single owner case. In this figure, the number of



(a) Single Owner, Single Cloud (b) Single Owner, 5 blocks/Cloud (c) Single Cloud, 5 blocks/Owner

Fig. 4. Comparison of Computation Cost of the Auditor (s = 50)

data blocks goes to 500 (i.e., the challenged data size equals to 500KByte), but it can illustrate the linear relationship between the computation cost of the auditor versus the challenged data size. From the Fig. 4(a), we can see that our scheme incurs less computation cost of the auditor than Zhu's IPDP scheme, when coping with large number of challenged data blocks.

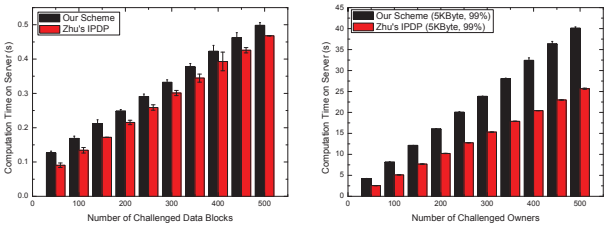
In real cloud storage systems, the data size is very large (e.g., petabytes), our scheme apply the sampling auditing method to ensure the integrity of such large data. The sample size and the frequency are determined by the service level agreement. From the simulation results, we can estimate that it requires 800 seconds to audit for 1GByte data. However, the computing abilities of the cloud server and the auditor are much more powerful than our simulation PC, so the computation time can be relatively small. Therefore, our auditing scheme is practical in large scale cloud storage systems.

Fig. 4(b) describes the computation cost of the auditor of the multi-cloud batch auditing scheme versus the number of challenged clouds. It is easy to find that our scheme incurs less computation cost of the auditor than Zhu's IPDP scheme, especially when there are a large number of clouds in the large scale cloud storage systems.

Because Zhu's IPDP does not support the batch auditing for multiple owners, in our simulation, we repeat the computation for several times which is equal to the number of data owners. Then, as shown in Fig. 4(c), we compare the computation cost of the auditor between our multi-owner batch auditing and the general auditing protocol which does not support the multi-owner batch auditing (e.g., Zhu's IPDP). Fig. 4(c) also demonstrates that the batch auditing for multiple owners can greatly reduce the computation cost. Although in our simulation the number of data owners goes to 500, it can illustrate the trend of computation cost of the auditor that our scheme is much more efficient than Zhu's scheme in large scale cloud storage systems that may have millions to billions of data owners.

6.2.2 Computation Cost of the Server

We compare the computation cost of the server versus the number of data blocks in Fig. 5(a) and the number of data owners in Fig. 5(b). Our scheme moves the computing loads



(a) Single Owner, Single Cloud (b) Single Cloud, 5 blocks/Owner

Fig. 5. Comparison of Computation Cost on the Serer (s = 50)

of the auditing from the auditor to the server, such that it can greatly reduce the computation cost of the auditor.

7 Related Work

To support the dynamic auditing, Ateniese *et al.* developed a dynamic provable data possession protocol [29] based on cryptographic hash function and symmetric key encryption. Their idea is to pre-compute a certain number of metadata during the setup period, so that the number of updates and challenges is limited and fixed beforehand. In their protocol, each update operation requires recreating all the remaining metadata, which is problematic for large files. Moreover, their protocol cannot perform block insertions anywhere (only append-type insertions are allowed). Erway *et al.* [22] also extended the PDP model to support dynamic updates on the stored data and proposed two dynamic provable data possession scheme by using a new version of authenticated dictionaries based on rank information. However, their schemes may cause heavy computation burden to the server since they relied on the PDP scheme proposed by the Ateniese.

In [23], the authors proposed a dynamic auditing protocol that can support the dynamic operations of the data on the cloud servers, but this method may leak the data content to the auditor because it requires the server to send the linear combinations of data blocks to the auditor. In [24], the authors extended their dynamic auditing scheme to be privacy-preserving and support the batch auditing for multiple owners. However, due to the large number of data tags, their auditing protocols will incur a heavy storage overhead on the server. In [25], Zhu *et al.* proposed

a cooperative provable data possession scheme that can support the batch auditing for multiple clouds and also extend it to support the dynamic auditing in [26]. However, it is impossible for their scheme to support the batch auditing for multiple owners. That is because parameters for generating the data tags used by each owner are different and thus they cannot combine the data tags from multiple owners to conduct the batch auditing. Another drawback is that their scheme requires an additional trusted organizer to send a commitment to the auditor during the batch auditing for multiple clouds, because their scheme applies the mask technique to ensure the data privacy. However, such additional organizer is not practical in cloud storage systems. Furthermore, both Wang's schemes and Zhu's schemes incur heavy computation cost of the auditor, which makes the auditing system inefficient.

8 Conclusion

In this paper, we proposed an efficient and inherently secure dynamic auditing protocol. It protects the data privacy against the auditor by combining the cryptography method with the bilinearity property of bilinear paring, rather than using the mask technique. Thus, our multi-cloud batch auditing protocol does not require any additional organizer. Our batch auditing protocol can also support the batch auditing for multiple owners. Furthermore, our auditing scheme incurs less communication cost and less computation cost of the auditor by moving the computing loads of auditing from the auditor to the server, which greatly improves the auditing performance and can be applied to large scale cloud storage systems.

References

[1] P. Mell and T. Grance, "The NIST definition of cloud computing," National Institute of Standards and Technology, Tech. Rep., 2009.

[2] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. H. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "A view of cloud computing," *Commun. ACM*, vol. 53, no. 4, pp. 50–58, 2010.

[3] T. Velte, A. Velte, and R. Elsenpeter, *Cloud Computing: A Practical Approach*, 1st ed. New York, NY, USA: McGraw-Hill, Inc., 2010, ch. 7.

[4] J. Li, M. N. Krohn, D. Mazières, and D. Shasha, "Secure untrusted data repository (sundr)," in *Proceedings of the 6th conference on Symposium on Operating Systems Design & Implementation*, Berkeley, CA, USA, 2004, pp. 121–136.

[5] G. R. Goodson, J. J. Wylie, G. R. Ganger, and M. K. Reiter, "Efficient byzantine-tolerant erasure-coded storage," in *DSN*. IEEE Computer Society, 2004, pp. 135–144.

[6] V. Kher and Y. Kim, "Securing distributed storage: challenges, techniques, and systems," in *StorageSS*, V. Atluri, P. Samarati, W. Yurcik, L. Brumbaugh, and Y. Zhou, Eds. ACM, 2005, pp. 9–25.

[7] L. N. Bairavasundaram, G. R. Goodson, S. Pasupathy, and J. Schindler, "An analysis of latent sector errors in disk drives," in *SIGMETRICS*, L. Golubchik, M. H. Ammar, and M. Harchol-Balter, Eds. ACM, 2007, pp. 289–300.

[8] B. Schroeder and G. A. Gibson, "Disk failures in the real world: What does an mttf of 1, 000, 000 hours mean to you?" in *FAST*. USENIX, 2007, pp. 1–16.

[9] M. Lillibridge, S. Elnikety, A. Birrell, M. Burrows, and M. Isard, "A cooperative internet backup scheme," in *USENIX Annual Technical Conference, General Track*. USENIX, 2003, pp. 29–41.

[10] Y. Deswarte, J. Quisquater, and A. Saidane, "Remote integrity checking," in *The Sixth Working Conference on Integrity and Internal Control in Information Systems (IICIS)*. Springer Netherlands, November 2004.

[11] M. Naor and G. N. Rothblum, "The complexity of online memory checking," *J. ACM*, vol. 56, no. 1, 2009.

[12] A. Juels and B. S. K. Jr., "Pors: proofs of retrievability for large files," in *ACM Conference on Computer and Communications Security*, P. Ning, S. D. C. di Vimercati, and P. F. Syverson, Eds. ACM, 2007, pp. 584–597.

[13] T. J. E. Schwarz and E. L. Miller, "Store, forget, and check: Using algebraic signatures to check remotely administered storage," in *ICDCS*. IEEE Computer Society, 2006, p. 12.

[14] D. L. G. Filho and P. S. L. M. Barreto, "Demonstrating data possession and uncheatable data transfer," *IACR Cryptology ePrint Archive*, vol. 2006, p. 150, 2006.

[15] F. Sebé, J. Domingo-Ferrer, A. Martínez-Ballesté, Y. Deswarte, and J.-J. Quisquater, "Efficient remote data possession checking in critical information infrastructures," *IEEE Trans. Knowl. Data Eng.*, vol. 20, no. 8, pp. 1034–1038, 2008.

[16] G. Yamamoto, S. Oda, and K. Aoki, "Fast integrity for large data," in *Proceedings of the ECRYPT workshop on Software Performance Enhancement for Encryption and Decryption*. Amsterdam, the Netherlands: ECRYPT, June 2007, pp. 21–32.

[17] M. A. Shah, M. Baker, J. C. Mogul, and R. Swaminathan, "Auditing to keep online storage services honest," in *HotOS*, G. C. Hunt, Ed. USENIX Association, 2007.

[18] C. Wang, K. Ren, W. Lou, and J. Li, "Toward publicly auditable secure cloud data storage services," *IEEE Network*, vol. 24, no. 4, pp. 19–24, 2010.

[19] K. Yang and X. Jia, "Data storage auditing service in cloud computing: challenges, methods and opportunities," *World Wide Web*, vol. 15, no. 4, pp. 409–428, 2012.

[20] G. Ateniese, R. C. Burns, R. Curtmola, J. Herring, L. Kissner, Z. N. J. Peterson, and D. X. Song, "Provable data possession at untrusted stores," in *ACM Conference on Computer and Communications Security*, P. Ning, S. D. C. di Vimercati, and P. F. Syverson, Eds. ACM, 2007, pp. 598–609.

[21] H. Shacham and B. Waters, "Compact proofs of retrievability," in *ASIACRYPT*, ser. Lecture Notes in Computer Science, J. Pieprzyk, Ed., vol. 5350. Springer, 2008, pp. 90–107.

[22] C. C. Erway, A. Küpçü, C. Papamanthou, and R. Tamassia, "Dynamic provable data possession," in *ACM Conference on Computer and Communications Security*, E. Al-Shaer, S. Jha, and A. D. Keromytis, Eds. ACM, 2009, pp. 213–222.

[23] Q. Wang, C. Wang, K. Ren, W. Lou, and J. Li, "Enabling public auditability and data dynamics for storage security in cloud computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 22, no. 5, pp. 847–859, 2011.

[24] C. Wang, Q. Wang, K. Ren, and W. Lou, "Privacy-preserving public auditing for data storage security in cloud computing," in *INFOCOM*. IEEE, 2010, pp. 525–533.

[25] Y. Zhu, H. Hu, G. Ahn, and M. Yu, "Cooperative provable data possession for integrity verification in multi-cloud storage," *Parallel and Distributed Systems, IEEE Transactions on*, pp. 1–14, 2011.

[26] Y. Zhu, H. Wang, Z. Hu, G.-J. Ahn, H. Hu, and S. S. Yau, "Dynamic audit services for integrity verification of outsourced storages in clouds," in *SAC*, W. C. Chu, W. E. Wong, M. J. Palakal, and C.-C. Hung, Eds. ACM, 2011, pp. 1550–1557.

[27] K. Zeng, "Publicly verifiable remote data integrity," in *ICICS*, ser. Lecture Notes in Computer Science, L. Chen, M. D. Ryan, and G. Wang, Eds., vol. 5308. Springer, 2008, pp. 419–434.

[28] G. Ateniese, S. Kamara, and J. Katz, "Proofs of storage from homomorphic identification protocols," in *ASIACRYPT*, ser. Lecture Notes in Computer Science, M. Matsui, Ed., vol. 5912. Springer, 2009, pp. 319–333.

[29] G. Ateniese, R. D. Pietro, L. V. Mancini, and G. Tsudik, "Scalable and efficient provable data possession," *IACR Cryptology ePrint Archive*, vol. 2008, p. 114, 2008.

[30] P. Ning, S. D. C. di Vimercati, and P. F. Syverson, Eds., *Proceedings of the 2007 ACM Conference on Computer and Communications Security, CCS 2007, Alexandria, Virginia, USA, October 28-31, 2007*. ACM, 2007.

ANDROID BASED MOBILE APPLICATION DEVELOPMENT and its SECURITY

Suhas Holla[#1], Mahima M Katti[#2]

[#] *Department of Information Science & Engg, R V College of Engineering Bangalore, India*

***Abstract*— In the advancing world of technology, Mobile applications are a rapidly growing segment of the global mobile market. Mobile applications are evolving at a meteor pace to give users a rich and fast user experience. In this paper, Android mobile platform for the mobile application development, layered approach and the details of security information for Android is discussed.**

Google released Android which is an open-source mobile phone operating system with Linux-based platform. It consists of the operating system, middleware, and user interface and application software. Certainly, Android is about to become the most widely used OS on mobile phones, but with Android comes a security vulnerability that few users take into account. On Android Market, where you can download thousands of applications for Android, anyone can upload their programs without having to submit them to careful security checks. This makes Android a prime target for computer criminals. In this paper, we discuss a layered approach for android application development where we can develop application which downloads data from the server. Also an Android Application Sandbox (AASandbox) which is able to perform both static and dynamic analysis on Android programs to automatically detect suspicious applications is also discussed.

***Keywords*— Android, application framework, android runtime, layered approach, AASandbox**

I. INTRODUCTION

Android is a new, next-gen mobile operating system that runs on the Linux Kernel. Android Mobile Application Development is based on Java language codes, as it allows developers to write codes in the Java language. These codes can control mobile devices via Google-enabled Java libraries. It is an important platform to develop mobile applications using the software stack provided in the Google Android SDK. Android mobile OS provides a flexible environment for Android Mobile Application Development as the developers can not only make use of Android Java Libraries but it is also possible to use normal Java IDEs. The software developers at Mobile Development India have expertise in developing applications based on Android Java Libraries and other important tools. Android Mobile Application Development can be used to create innovative and dynamic third party applications. Mobile Development India has worked extensively on projects ranging from gaming software, organizers, media players, picture editors to go-cart devices and more.

II. BACKGROUND STUDY

The platform was officially announced and the SDK tools were available in October 2008. Currently there is only one mobile phone that runs the Android OS, the G1 from T-Mobile. According to the official Android website (Android 2008) the platform is based into the four core features as shown in the Fig 1:

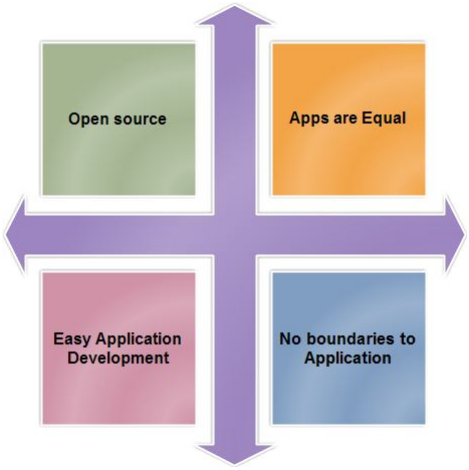


Fig. 1 Four core features of the android platform

A. Application Fundamentals

Android applications are written in Java programming language. However, it is important to remember that they are not executed using the standard Java Virtual Machine (JVM). Instead, Google has created a custom VM called Dalvik which is responsible for converting and executing Java byte code. All custom Java classes must be converted (this is done automatically but can also be done manually) into a Dalvik compatible instruction set before being executed into an Android operating system. Dalvik VM takes the generated Java class files and combines them into one or more Dalvik Executable (.dex) files. It reuses duplicate information from multiple class files, effectively reducing the space requirement (uncompressed) by half from a traditional .jar file. Dalvik was

created to support the nature of lightweight mobile operating systems require because of the limited hardware capabilities compared to conventional desktops or laptops.

B. Android Platform overview

Android is a software stack for mobile devices that includes an operating system, middleware and key applications. The Android SDK provides the tools and APIs necessary to begin developing applications on the Android platform using the Java programming language [3]. Android based on Linux version 2.6. The system services such as security, memory management, process management are controlled by Linux. Fig 2 shows android architecture.

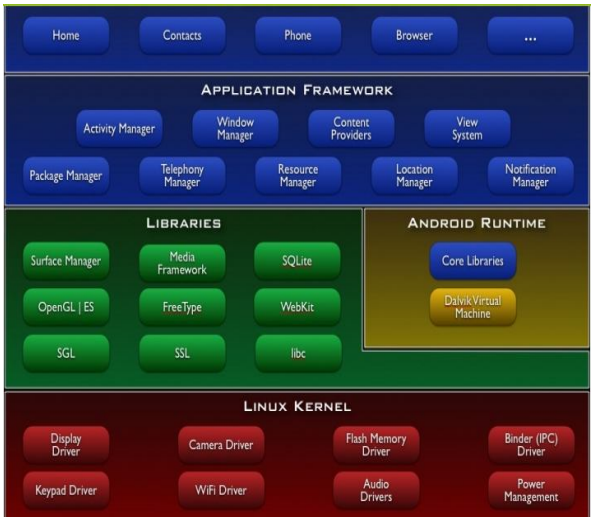


Fig. 2 Architecture of android [1]

C. Developing Android Applications

The Android SDK provides an extensive set of application programming interfaces (APIs) that is both modern and robust. Android handset core system services are exposed and accessible to all applications. When granted the appropriate permissions, Android applications can share data among one another and access shared resources on the system securely [5]. Android applications are written in Java programming language.

D. Application Framework

By providing an open development platform, Android offers developers the ability to build extremely rich and innovative applications. Developers are free to take advantage of the device hardware, access location information, run background services, set alarms, add notifications to the status bar, and much, much more.

Developers have full access to the same framework APIs used by the core applications. The application architecture is designed to simplify the reuse of components; any application can publish its capabilities and any other application may then make use of those capabilities (subject to security constraints enforced by the framework). This same mechanism allows components to be replaced by the user.

Underlying all applications is a set of services and systems, including:

- A rich and extensible set of Views that can be used to build an application, including lists, grids, text boxes, buttons, and even an embeddable web browser
- Content Providers that enable applications to access data from other applications (such as Contacts), or to share their own data
- A Resource Manager, providing access to non-code resources such as localized strings, graphics, and layout files
- A Notification Manager that enables all applications to display custom alerts in the status bar
- An Activity Manager that manages the lifecycle of applications and provides a common navigation backstack.

E. Android Runtime

Android includes a set of core libraries that provides most of the functionality available in the core libraries of the Java programming language [5]. Every Android application runs in its own process, with its own instance of the Dalvik virtual machine. Dalvik has been written so that a device can run multiple VMs efficiently. The Dalvik VM executes files in the Dalvik Executable (.dex) format which is optimized for minimal memory footprint. The VM is register-based, and runs classes compiled by a Java language compiler that have been transformed into the .dex format by the included "dx" tool. The Dalvik VM relies on the Linux kernel for underlying functionality such as threading and low-level memory management.

III. Layered Approach for Application Development

In this paper we suggest layered approach for android application development. This can be used for web based application development.

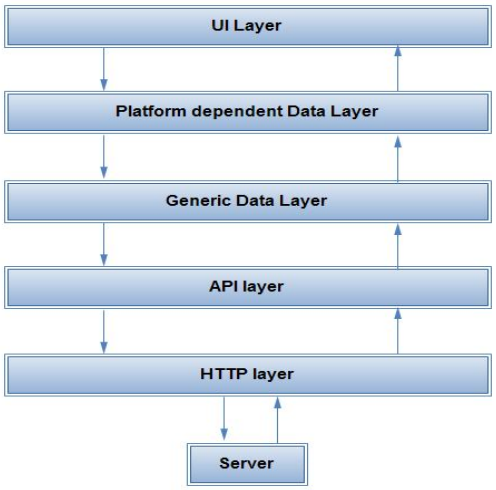


Fig. 3 Layered architecture

Figure 3 shows the layered approach for the android application development. The lowest level is HTTP layer which is responsible for sending HTTP get and post requests to the server and receiving the response. Next layer is API layer. This is for parsing the response from the server and formulating the query and passing it to the HTTP layer. The API layer gets the response string from the HTTP layer and parses the string. It also helps in extracting the necessary fields and passes it to the data layer. The Generic Data layer contains the components that include designing business layers and implementing functionalities like caching, exceptional management, logging and validation. Next is platform dependent data layer which takes the data from the API layer and use it. It stores the data in the platform dependent way. Some classes like Adapter, Listview etc store the data dependent on the platform. Last one the UI layer. This helps in showing the data to the user and manages user interactions. It has two components user interface components and user process components. User interface components provide a way for users to interact with the application. User process components synchronize and organize user interactions. UI layer is responsible for views in android. It has Views, buttons, layouts etc.

A. The application model

In Android's application model [1], an application is a package of components, each of which can be instantiated and run as necessary (possibly even by other applications). Components are of the following types [5]:

Activity components form the basis of the user interface; usually, each window of the application is controlled by some activity. **Service** components run in the background, and remain active even if windows are switched. Services can expose interfaces for communication with other applications. **Receiver** components react asynchronously to messages from other applications. **Provider** components store data relevant to the application, usually in a database. Such data can be shared across applications [3].

Consider, e.g., an online photo viewing application for an Android based phone. This application may have several components. There are activities for viewing the photos on the phone in the form of grid or list. There may be a service for downloading a photo in the background. There may be receivers for pausing a application when a call comes in, and for restarting the application when the call ends. The application should not affect the high priority functionality of the device like incoming call, incoming sms, battery low indication etc. Finally, there may be a provider for storing the photos and its details on the phone.

B. Component classes and methods

The Android SDK has a base class for each type of component (Activity, Service, Receiver, and Provider), with callback methods that are invoked at various points in the life cycle of the associated component. Each component has a life cycle. Each component of an application is defined by extending one of the base classes, and overriding the methods in that class. In particular:

- The Activity class has methods that are run when activity is created, or activity calls some other activity, or returns to the activity.
- The Service class has methods that are run when the service is started, or some component binds to this service or even combination of both.
- The Receiver class has a method that is run when a message is sent to this receiver.
- The Provider class has methods to delete, query and update the data stored by this provider.

C. Component classes and methods

The Google Android mobile phone platform is one of the most anticipated smartphone operating systems. Smart phones can be used in place of Computers/Laptops. As mobile devices attain increasing capabilities, there are many more opportunities for novel applications development. Recent development of mobile application development has reached a high demand on today's cellular market. Android defines a new component-based framework for developing mobile applications, where each application is comprised of different numbers and types of components. Activity components are the basis of the user interface; each screen presented to the user is a different Activity [6]. Service components provide background processing that continues even after its application loses focus. Content Provider components share information in relational database form. SQLite is embedded into android which supports relational database. For instance, the system includes an application with a Content Provider

creates a new user profile associated with the application. Each application runs as a different user, with its own private files on the file system, a user ID, and a secure operating environment. The application executes in its own process with its own instance of the Dalvik VM and under its own user ID on the operating system.

B. Explicitly Defined Application Permissions

When an Android requires explicitly defined application permissions in the manifest file. To access shared resources on the system, Android applications register for the specific privileges they require. While developing the application required permissions should be specified in Android manifest file. For example if the phone vibration functionality is required then it must be specified in the android manifest file. While installing the application it shows the list of resources that the application is going to access. Some of these privileges enable the application to use phone functionality to make calls, access the network, and control the camera and other hardware sensors. Applications also require permission to access shared data containing private and personal information such as user preferences, user's location, and contact information. Applications might also enforce their own permissions by declaring them for other applications to use. The application can declare any number of different permission types, such as read-only or read-write permissions, for finer control over the application.

C. Limited Ad-Hoc Permissions

Content providers might want to provide some on-the-fly permissions to other applications for specific information they want to share openly. This is done using ad-hoc granting and revoking of access to specific resources using Uniform Resource Identifiers (URIs). URIs points to specific data assets on the system, such as MediaStore, Contacts, CallLog etc. Here is an example of a URI that provides the phone numbers of all contacts:

content://contacts/phones.

D. Application Signing for Trust Relationships

All Android applications packages are signed with a certificate, so users know that the application is authentic. The private key for the certificate is held by the developer. This helps establish a trust relationship between the developer and the user. It also allows the developer to control which applications can grant access to one another on the system. No certificate authority is necessary; self-signed certificates are acceptable.

VI. CONCLUSION

With the vigorous development through Android, mobile applications have been widely used on the various mobile devices. Android mobile applications are evolving at a meteor pace to give a rich and fast user experience. The maturity of the hardware and software platforms of mobile devices and the promotion of the Mobile Internet have brought a great opportunity to the migration of the web applications to mobile platforms. In case of the security, Static analysis scans the software for malicious patterns without installing it. Dynamic analysis executes the application in a fully isolated environment, i.e. sandbox, which intervenes and logs low-level interactions with the system for further analysis. Both the sandbox and the detection algorithms can be deployed in the cloud, providing a fast and distributed detection of suspicious software in a mobile software store akin to Google's Android Market. The ultimate goal is to protect the mobile applications from the malicious attributes and safeguard the interests of Android mobile users. With the mobile capabilities, the Internet connection capabilities and complete software platforms available, the future of mobile web application appear limitless.

VII. FUTURE WORK

The era of mobile web application has just started, and there is a long way for it to march. Development of mobile web application will be emphasized on following aspects:

1) More and more sensors will be added to mobile phones, so new APIs to use those capabilities will bring brand new applications to users.

2) Multimedia capabilities will be enhanced and engine will support more types of multimedia such as flash and svg .

3) The dedicated Integrated Development Environment (IDE) will be improved to accelerate the applications' development. Visualization programming and JavaScript debugging will be the most important functions of the IDE.

REFERENCES

[1] What is android? http://developer.android.com/guide/basics/what-is-android.html

[2] 3G Mobile Terminal Development Trend of the operating system [M/OL]. http://pda.c114.net/32/c4948.html, 2007

[3] Android Architecture 2010[R/OL]. http://www.cnmsdn.com/html/201003/1268713218ID2058_2 .html.

[4] Static detection of malicious code in executable programs by J. Bergeron, M. Debbabi, J. Desharnais, M. M. Erhioui, Y. Lavoie, and N. Tawbi.

[5] Android Official Website (2008)—"Android | Official Website", <http://www.android.com/>.

[6] An Android Application Sandbox System for Suspicious Software Detection, by Thomas Bl¨asing, Leonid Batyuk, Aubrey-Derrick Schmidt, Seyit Ahmet Camtepe, and Sahin Albayrak

[7] www.blackhat.com

Audit Automation for Implementing Continuous Auditing: Principles and Problems

Michael G. Alles
Alexander Kogan
Miklos A. Vasarhelyi

Rutgers Business School

Department of Accounting,
Business Ethics & Information Systems
180 University Ave
Newark, NJ 07102

Version: August 20, 2008[*]

Abstract: When implementing continuous auditing, experience indicates that auditors will likely attempt to first automate the processes that they already use, are comfortable with and are already accepted for external auditing and reporting purposes rather than trying to start from scratch, especially when dealing with audits of ongoing operations. However, because of the experience of low productivity and failed expectations with prior technology implementations that gave rise to the argument for business process reengineering in Hammer's (1990) article, "Don't Automate, Obliterate" special care needs to be taken when change is brought about by automation. As we argue in this paper, not only must audit automation be undertaken systematically, it also has to incorporate reengineering in the more limited sense of first transforming manual audit processes to facilitate their automation. This is not full blown reengineering of the clean sheet sort, but this hybrid approach is one that is more manageable—and marketable—from a change management perspective, and more likely to lead to a positive outcome. The key for avoiding the potential downsides of automation, though, is to have a clear understanding of what audit automation is trying to achieve and follow a methodical procedure to achieve those goals.

Keywords: audit automation, continuous auditing, internal audit, audit systems.

[*] Comments welcome. Please address them to kogan@rbsmail.rutgers.edu.

1. Introduction

Alles et al (2006) described a feasibility study undertaken by the IT Internal Audit department at Siemens Corporation, working with the authors, to create a continuous auditing (CA) system by automating the largely manual audit of its SAP systems. Since that initial study, the field of continuous auditing has rapidly developed, with vendors offering sophisticated IT products that facilitate CA implementation, as well as many other firms having begun to develop homegrown CA processes. This paper builds on the authors experience with such vendors and firms, as well work we undertook with Siemens on its second generation CA implementation. While the details of that particular case study is described elsewhere (Teeter and Brennan, 2008), in this paper we step back to draw general conclusions about the challenges that auditors will face when automating existing audit procedures for a CA environment, as well as the opportunities that they now have with new CA-enabling technologies.

As with Alles et al. (2006), our focus is on automation of an existing, predominantly manual audit process. While in years to come new CA audits may be created with a blank sheet approach, the experience of much technology implementation from mini-computers to ERP suggests that the change process is likely to be incremental rather than disruptive. Hence, as at Siemens, auditors will likely attempt to first automate the processes that they already use, are comfortable with and are already accepted for external auditing and reporting purposes rather than trying to start from scratch, especially when dealing with audits of ongoing operations. Moreover, audit standards have been largely written for a world in which technology may be an enabler, rather than the driver of audit processes as it is in CA, which again implies that the current need is for an understanding of automation as the primary mechanism used to bring about CA.

At the same time, precisely because of the experience of low productivity and failed expectations with prior technology implementations that gave rise to the argument for business process reengineering in Hammer's (1990) article, "Don't Automate, Obliterate" special care needs to be taken when change is brought about by automation. As we argue in this paper, not only must audit automation be undertaken systematically, it also has to incorporate reengineering in the more limited sense of first transforming manual audit processes to facilitate their automation. This is not full blown reengineering of the "throw

away the [manual] rule book" sort, but this hybrid approach is one that is more manageable—and marketable—from a change management perspective, and more likely to lead to a positive outcome. The key for avoiding the potential downsides of automation, though, is to have a clear understanding of what audit automation is trying to achieve and follow a methodical procedure to achieve those goals.

We begin by considering the case for audit automation and its relation to continuous auditing.

2. Drivers and Objectives of Audit Automation

Half a century has passed since the original utilization of computers in business. Even in those early days the tremendous potential impact of automation on accounting and auditing was understood and brought to the attention of the profession and academia (Keenoy, 1958). What was in the beginning a limited scope deployment focused on carefully selected business areas (such as payroll processing and inventory control) and utilized primarily by large enterprises, has since become ubiquitous for most business entities, and is now inseparable from doing business. Most business processes today are automated to various degrees, and businesses continue to invest in maintaining and expanding this automation through the acquisition of computer and telecommunication technologies and various enterprise systems, such as enterprise resource planning, data warehousing, supply chain management, and customer relationship management.

Automation of business processes has inevitably led to changes in auditing procedures and standards. Starting with the original release in 1973 of SAS 3 ("The Effects of EDP on the Auditor's Study and Evaluation of Internal Controls"), the utilization of modern information technology has made its way into the audit process, prompted by growing availability and decreasing cost of personal computing and office automation software (word processors and spreadsheets), as well as generic statistical software and dedicated computer-assisted audit techniques, such as Audit Command Language. These early deployments demonstrated the potential of IT in making labor-intensive repetitive audit work more efficient. The potential impact of audit automation on the changes in the audit process was originally researched almost quarter century ago by Vasarhelyi (1984). Recent empirical studies (Banker et al., 2002) have shown that IT-enabled audit automation indeed leads to significant productivity gains.

automation involved primarily CCM, the general principles and problems of audit automation generalized from that experience apply to both aspects of CA.

3. Audit Automation Change Management

The audit profession is inherently conservative given that its entire value added comes from the auditor's credible claims of objectivity and reliability. As a consequence, auditing processes, even more so than other business processes, have a tremendous amount of inertia. It follows that any audit automation project, as with any major change initiative in such circumstances, will have numerous barriers to change to overcome. This is why it is critical to ground audit automation projects in sound business process change methodologies developed in the management literature as a result of extensive experience with large scale IT implementations (see e.g., Davenport and Short, 1990; Wastell et al., 1994; Kettinger et al., 1997; Reijersa and Liman Mansar, 2005).

As research on ERP implementation success factors has convincingly shown (Umble et al., 2003; Botta-Genoulaz et al., 2005), for an automation project to even get launched, let alone succeed, senior executive champions have to take ownership of the project, both at the internal audit level, and at their reporting level in the C-suite or the audit committee. In the case of Siemens the champion that brought the authors on board was the then head of the IT Internal Audit Department. The fact that we are increasingly coming across executives at firms with titles such as "Associate Director, Continuous Assurance" (in the case of BD Corporation) indicates that such champions are becoming institutionalized in firms as CA goes mainstream.

The first critical task of audit automation champions will be to identify and engage project stakeholders. In addition to internal auditors, these stakeholders will include business process owners and IT personnel. Again, the use of such multifunctional teams is a standard recommendation of change management theory, but in the case of audit automation the problem is compounded by the need of internal audit to be aware of the needs of the external auditor, while also balancing the demands of the IT process owners and line managers. The composition of audit automation teams must reflect the multi-faceted nature of the task at hand.

The reason for having a high powered team with a senior level champion is obvious when considering the complexity inherent in automating audit processes initially designed to be done largely manually. In our experience, even very experienced auditors differ in how such procedures are carried out in practice, which translates into differences in how to transform the process into an automated one, what the objective of the process should be and how much weight should be placed on a particular process or on a possible compensating control.

A powerful way of increasing the quality and reliability of audit automation results would be to diversify risk and bring out differences by utilizing duplicate audit automation teams, and then comparing the resulting automated audit programs. Resolving the inevitable disagreements between the duplicate teams would greatly improve the final automated program. While this approach is often utilized in academic research (e.g., to make sure that human responses are coded properly), we have yet to come across any instance in which such a procedure has been adopted. It is simply too expensive, both in terms of human resources and time to be feasible for the vast majority of enterprises. Therefore, alternative measures have to be utilized to assure the quality of the automated audit program both during the automation process and to verify the completed product.

One such alternative is for the automated audit procedure developed by the automation team to be verified independently by experienced auditors who took no part in developing it. A vitally important check on the audit automation process is the need to satisfy the external auditor, and to retain their reliance on the internal audit process. As, in accordance with SAS 65 ("The Auditor's Consideration of the Internal Audit Function in an Audit of Financial Statements"), the external auditor evaluated the quality and effectiveness of the original manual audit process, such evaluation can encompass the automation process, the finished automated audit program, or ideally, both. In either case, demonstrating that the automation team followed a systematic procedure is an essential element in satisfying the external auditor.

4. Formalizing and Reengineering the Audit Program

As Alles et al. (2006) indicated, before audit procedures can be automated, they must first be formalized: *"Automation requires formalization of audit procedures. Approved audit programs are not*

contracts. At the same time, the possibility of formalization is often underestimated, and when an earnest effort is made to formalize audit procedures, the results often exceed the most optimistic expectations. Alles et al. (2006) concluded this much in their study of Siemens first generation CA implementation, but importantly, Teeter and Brennan (2008) indicated that recent advances in CA-enabling software, such as Approva, have increased the scope for formalization and audit automation.

Siemens' internal audit methodology for SAP facilities involves the carrying out of several hundred of "audit action sheets" by internal auditors at the auditee site. Alles et al. (2006) indicated that about 25% of the audit actions could be fully automated due to their deterministic nature. But Teeter and Brennan (2008), in their study of the CA initiative based on an Approva system foundation concluded that about 68% of the actions could be automated to some extent. Considering that some of these automated steps would be performed in a daily monitoring mode (as opposed to the 18 to 24 month cycle of SAP audits) the strength of its evidence would be much stronger and conceivably could replace much of the residual 32% non-automated evidence. Such replacements have been shown in some past experience to be the main benefit of audit automation (Fischer, 1996).

5. Baseline Monitoring ("Baselining")

Baseline monitoring or "baselining" for short is a well established procedure in configuration management and IT security, defining the "what should be" state that is used as a benchmark against which the current state can be compared. As applied to enterprise systems, baseline is defined as a set of system configuration and business process settings at a given, reference point of time.

Baselining is facilitated in audit automation through the use of the "snapshot" feature of audit automation software (such as Approva), which does precisely what the name suggests (albeit only for the SAP tables that the program monitors). The use of baselining allows the automation of audit procedures that would be difficult to formalize explicitly, by allowing for a simple before/after comparison. The price to pay for this approach towards automation is the extensive manual effort to initially verify all the values in the baseline to make sure that they are appropriate. This verification relies on human judgment typically supported by manual techniques such as interviewing, investigations and observations. However,

Datar (2004) state: *"Interactive controls are particularly important in control theory because it is they that tell senior managers when they need to change strategy—and so, when they have to alter the belief, boundary and diagnostic controls that put that strategy into practice. In other words, interactive controls make the entire control architecture dynamic, enabling it to evolve as underlying conditions change, or as core assumptions are proved invalid."*

The application to baselining is to draw the analogy of the baseline to a boundary control and to develop an analytic engine for alarms as an interactive control that would indicate when exceptions are being caused by a change in the underlying operations, thus necessitating an evolution of the automated audit, as opposed to an anomaly caused by inappropriate behavior by the auditee. In other words, baselining and their technological enabler, the "snapshot" feature of monitoring software, are simply tools, means towards an end. Putting them to work as an automated audit procedure requires that they be overlaid with an effective control framework that will enable their transformation into a dynamic monitoring process. It is possible that combining Simons' framework with the very powerful snapshot capabilities of modern audit enabling software will lead to the development of a highly capable CA system that goes beyond first-generation audit automation.

However, it needs to be reiterated that the initial manual verification of baseline values is a critical stage of audit automation. Any mistake made at this stage will be leveraged since the system will automatically perpetuate the mistake indefinitely. This is an indication of the different set of risks that can arise in automated systems. While in manual auditing there is always a chance that a mistake made during a particular audit cycle can be corrected during subsequent periods, in baselining there is only one chance during the initial verification stage to get things right.

Another critically important issue is the security of baseline—both in its definition and its current values. If one can compromise the security of the baseline and manipulate the definition of the baseline, say by removing from it certain parameters or by changing certain values in the baseline, then one can potentially open gaping holes in the system without anybody ever noticing. Thus, securing the baseline must be a well-developed feature of an automated auditing system.

6. Architecture of Automated Auditing

The other side of the issues discussed above is the necessity to protect the EAM or mobile agent auditing code against possible manipulation by the enterprise platform. Given that the superuser privileges for the enterprise system are held by the enterprise IT personnel, the integrity of the audit code processing is always in question since it is the objective of this code to check on the enterprise system and its personnel. This problem has been discussed in the literature under the name of the *malicious host problem* (Jansen and Karygiannis, 1999; Claessens et al., 2003), and it is considered to be extremely difficult (if not infeasible). While there have been proposed numerous ways of dealing with it (see e.g., Stengel et al, 2005; Futoransky et al, 2006; Shao and Zhou, 2006; Topaloglu and Bayrak, 2008), and there are some quite complex ways of detecting the problem, no solution has been universally accepted as being able to prevent the host from interfering with code's execution.

The extreme difficulty (if not impossibility) of protecting the EAM or mobile agent auditing code from possible manipulation by the enterprise platform puts in question the integrity of results provided by this auditing code. This lack of trust in the audit results outweighs the benefits of the resident code described above, and serves as one of the critical reasons for basing automated auditing architecture on remote monitoring of enterprise systems.

7. Handling, Evaluation and Integration of Audit Evidence

While it is the ultimate objective of any business enterprise to have a totally reliable control system that will not have any exceptional events or anomalous situations, this ideal is never achieved, and the auditing system will be generating alarms caused by anomalies and exceptions. These alarms will be delivered by automatic means (e-mail, instant and wireless messaging) to the appropriate auditors and enterprise personnel responsible for resolving them. While the automated auditing system keeps track of each event, it is essential to have an automated closed loop process for capturing information about the corrective actions and assuring that these actions resolve the underlying problem.

As the results of resolving exceptional events and anomalous situations identify various control failures of the enterprise system, the auditing system should have a built-in mechanism for evaluating how significant these failures are, and making these evaluations available in a timely manner to the relevant stakeholders (auditors and upper management). To make such automatic evaluations possible, the procedures in the automated auditing

system have to be organized in accordance with the enterprise risk model to associate appropriate risks to various control failures.

While individual evaluations of control failures are no doubt important, large enterprises in particular would be interested in aggregating audit evidence to see the "big picture" of current enterprise exposure. The development of sound theoretical methodology for measuring internal control performance and aggregating audit evidence that would be practically applicable in modern large enterprises presents serious challenges. Theoretical studies of internal controls design and evaluation undertaken over the years (Cushing, 1974; Srinidhi, 1988; Vasarhelyi and Srinidhi, 1989; Knorr and Stormer, 2001) can provide a foundation for future practical developments which are yet to come. In the meantime, various ad hoc solutions and simplifying assumptions can be utilized to build a continuous auditing dashboard that in real time provides an aggregate view of enterprise control problems.

8. Software for Audit Automation

While it is certainly possible to design, develop and implement a custom-made automated auditing system in house, the expense and expertise requirements of such a project make it prohibitively expensive, if not outright infeasible, for the vast majority of cases. It is therefore not surprising that there is an emerging industry of packaged software developed to support audit automation or at least some of its aspects.

A convenient way of categorizing the current software offerings is in accordance with the breakdown of CA as consisting of CCM and CDA. While the vendors are attempting to integrate in their packages as many features as possible, they still typically exhibit strength in one of the two components. The well-established CAATs vendors ACL and CaseWare IDEA have extended their products to position them as continuous monitoring solutions. ACL in particular has invested significant efforts into providing what they call "continuous controls monitoring" solutions. Despite the name, in the terminology of this paper these solutions should be categorized as CDA since the substance of their tests is transaction verification and analysis focused on making inference about the functioning of controls (as opposed to direct tests of controls through monitoring of their settings). A relative

functionality that would bring these solutions closer to the fully developed CCM and/or CDA systems.

Automating a manual audit program requires a significant startup expense. This fixed cost may become a significant hurdle in the way of audit automation if an enterprise has no way of amortizing this cost over different enterprise units, since automation of highly specific audit procedures for different enterprise units can incur prohibitive costs. Automation will be scalable across the enterprise only if the repetitive audit procedure automation costs are eliminated.

There are a number of strategies for making audit automation scalable. The most immediate one is the parameterization of automated audit procedures. Given the expected significant homogeneity of enterprise business processes, it is likely that one can make automated audit procedures sufficiently generic by introducing various parameters describing systems, processes and business artifacts. Then the implementation of automated audit in additional enterprise units can be reduced to properly configuring the already developed system by assigning the appropriate parameter values.

In addition to parameterization, scalability of the audit program can be enhanced through hierarchical structuring of automated audit procedures – from the most generic audit procedures applicable across the enterprise to the more specific ones for major units and subunits. The feasibility of such structuring is enabled by the natural hierarchy of business enterprises and the risk-based top-down approach towards audit program development. This will also facilitate audit program maintenance through hierarchical updates: given a change in the processes at a certain node of the enterprise hierarchy, only the audit procedures in the hierarchical sub-tree rooted at this node will have to be reviewed and, possibly, revised.

9. Securing Continuous Auditing

Based on the analysis above, it is likely that an automated auditing system will be implemented as a MCL hosted on its own platform. To assure the integrity of its results, this system must be thoroughly secured. One of the issues that critically affect this system security is the control of the continuous auditing software, and its associated hardware,

which can be either under the authority of the auditee's IT department, or under the internal auditors themselves. Although the latter is intrinsically more secure, there are numerous practical matters that can favor the former. While we have come across instances in which the internal auditors have their own CA software (albeit, running on the firm's general IT systems), the cost and complexity of software such as Approva makes it far more likely to be entirely run by the firm's IT department (or outsourced to the vendors or third parties) with access provided to the auditors. In this case, one has to pay particular attention to access security.

Logical access security of the auditing system is even more critical since any compromise of the system can potentially be used to cover up for undesirable events in the enterprise system. As Alles et al. (2002, 2004) argued, automated audit systems are in one sense more vulnerable than manual systems due to the lack of "another-pair-of-eyes" controls and the leveraging of a mistake—intentional or otherwise—every time the CA system runs. Maintaining logical security is particularly problematic since it requires advanced system management IT skills which may not be easily available in internal auditing. The super-user privileges in the auditing system are figuratively speaking the "keys to the kingdom", and their safeguarding requires utmost attention, which is why this is a key control for SOX 404 certification.

To mitigate this exposure, comprehensive logging of all super-user activities should also be implemented and constantly monitored by the internal auditors, as Alles et al. (2004) proposed. An important control over the security of the auditing system consists in exporting its settings and cryptographic check-sums of its code to an external storage facility or/and non-volatile storage medium. Then these setting can be imported back into the CA system as needed, and the cryptographic check-sums of the system code can be recalculated to verify the integrity of the CA system. This is obviously a de-facto tertiary monitoring process, and the administration of this process should be done by dedicated internal audit personnel.

10. Concluding Remarks

The practice of audit automation will be strongly influenced by the ongoing software development and maturing of the field of GRC. AMR Research projects that spending on

quality of work the automated auditing systems are providing. It is likely that it will also result in significant changes in the nature of audit procedures performed by the external auditor, and these changes in turn may lead to structural changes in audit engagements, and may, with time, reshape the external audit as we know it today.

References

1. Alles, M.A., and Datar, S. 2004. Cooking the Books: A management-control perspective on financial accounting standard setting and the section 404 requirements of the Sarbanes-Oxley Act. *International Journal of Disclosure and Governance,* Vol. 1, No. 2, 119–137.
2. Alles, M.G., A. Kogan, M.A. Vasarhelyi. 2002. Feasibility and economics of continuous assurance. *Auditing: A Journal of Practice and Theory*, Vol. 21, No. 1 (March), 125–138.
3. Alles, M.G., A. Kogan, M.A. Vasarhelyi. 2004. Restoring Auditor Credibility: Tertiary Monitoring and Logging of Continuous Assurance Systems. *International Journal of Accounting Information Systems*, Vol. 5, No. 2 (June), 183–202.
4. Alles, M.G., G. Brennan, A. Kogan, M. A.Vasarhelyi. 2006. Continuous Monitoring of Business Process Controls: A Pilot Implementation of a Continuous Auditing System at Siemens. *International Journal of Accounting Information Systems*, Vol.7, 137–161.
5. Alles, M.G., A. Kogan, M.A. Vasarhelyi, J. Wu. 2008a. *Continuous Data Level Auditing Using Continuity Equations.* Unpublished working paper, Rutgers Business School.
6. Alles, M.G., A. Kogan, M.A. Vasarhelyi. 2008b. Putting continuous auditing theory into practice: Lessons from two pilot implementations. *Journal of Information Systems*, forthcoming.
7. Banker, R. D., H. Chang, and Y. Kao. 2002. Impact of information technology on public accounting firm productivity. *Journal of Information Systems* 16 (2): 209–222.
8. Bedard, J. C., D. R. Deis, M. B. Curtis, J. G. Jenkins. 2008. Risk monitoring and control in audit firms: A research synthesis. *Auditing: A Journal of Practice & Theory*, Vol. 27, No. 1 (May), 187–218.
9. Bell, T.B., F. Marrs, I.Solomon, H. Thomas. 1997. *Auditing Organizations Through a Strategic-Systems Lens: The KPMG Business Measurement Process.* KPMG, Montvale, NJ.
10. Botta-Genoulaz, V., P.-A. Millet, B. Grabot. 2005. A survey on the recent research literature on ERP systems. *Computers in Industry*, Vol. 56, No. 6 (Aug.), 510–522.
11. Claessens, J., B. Preneel, J. Vandewalle. 2003. (How) can mobile agents do secure electronic transactions on untrusted hosts? A survey of the security issues and the current solutions. *ACM Transactions on Internet Technology*, Vol. 3, No. 1 (February), 28–48.
12. Cooke, N. J. 1994. Varieties of knowledge elicitation techniques. International *Journal of Human-Computer Studies*, Vol. 41, No. 6 (December), 801–849.

Authorized Public Auditing of Dynamic Big Data Storage on Cloud with Efficient Verifiable Fine-Grained Updates

Chang Liu, Jinjun Chen, *Senior Member, IEEE*, Laurence T. Yang, *Member, IEEE*, Xuyun Zhang, Chi Yang, Rajiv Ranjan, and Ramamohanarao Kotagiri

Abstract—Cloud computing opens a new era in IT as it can provide various elastic and scalable IT services in a pay-as-you-go fashion, where its users can reduce the huge capital investments in their own IT infrastructure. In this philosophy, users of cloud storage services no longer physically maintain direct control over their data, which makes data security one of the major concerns of using cloud. Existing research work already allows data integrity to be verified without possession of the actual data file. When the verification is done by a trusted third party, this verification process is also called data auditing, and this third party is called an auditor. However, such schemes in existence suffer from several common drawbacks. First, a necessary authorization/authentication process is missing between the auditor and cloud service provider, i.e., anyone can challenge the cloud service provider for a proof of integrity of certain file, which potentially puts the quality of the so-called 'auditing-as-a-service' at risk; Second, although some of the recent work based on BLS signature can already support fully dynamic data updates over fixed-size data blocks, they only support updates with fixed-sized blocks as basic unit, which we call coarse-grained updates. As a result, every small update will cause re-computation and updating of the authenticator for an entire file block, which in turn causes higher storage and communication overheads. In this paper, we provide a formal analysis for possible types of fine-grained data updates and propose a scheme that can fully support authorized auditing and fine-grained update requests. Based on our scheme, we also propose an enhancement that can dramatically reduce communication overheads for verifying small updates. Theoretical analysis and experimental results demonstrate that our scheme can offer not only enhanced security and flexibility, but also significantly lower overhead for big data applications with a large number of frequent small updates, such as applications in social media and business transactions.

Index Terms—Cloud computing, big data, data security, provable data possession, authorized auditing, fine-grained dynamic data update

1 Introduction

Cloud computing is being intensively referred to as one of the most influential innovations in information technology in recent years [1], [2]. With resource virtualization, cloud can deliver computing resources and services in a pay-as-you-go mode, which is envisioned to become as convenient to use similar to daily-life utilities such as electricity, gas, water and telephone in the near future [1]. These computing services can be categorized into Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS) and Software-as-a-Service (SaaS) [3]. Many international IT corporations now offer powerful public cloud services to users on a scale from individual to enterprise all over the world; examples are Amazon AWS, Microsoft Azure, and IBM SmartCloud.

Although current development and proliferation of cloud computing is rapid, debates and hesitations on the usage of cloud still exist. Data security/privacy is one of the major concerns in the adoption of cloud computing [3], [4], [5]. Compared to conventional systems, users will lose their direct control over their data. In this paper, we will investigate the problem of integrity verification for big data storage in cloud. This problem can also be called data auditing [6], [7] when the verification is conducted by a trusted third party. From cloud users' perspective, it may also be called 'auditing-as-a-service'. To date, extensive research is carried out to address this problem [6], [7], [8], [9], [10], [11], [12], [13], [14], [15]. In a remote verification scheme, the cloud storage server (CSS) cannot provide a valid integrity proof of a given proportion of data to a verifier unless all this data is intact. To ensure integrity of user data stored on cloud service provider, this support is of no less importance than any data protection mechanism deployed by the cloud service provider (CSP) [16], no matter how secure they seem to be, in that it will provide the verifier a piece of direct, trustworthy and real-timed intelligence of the integrity of the cloud user's data through a challenge request. It is especially recommended that data auditing is to be conducted on a regular basis for the

- *C. Liu is with the School of Comput. Sci. and Tech., Huazhong Uni. of Sci. and Tech., China, and also with the Faculty of Eng. and IT, Uni. of Tech., Sydney, Australia. E-mail: changliu.it@gmail.com.*
- *J. Chen, X. Zhang, and C.Yang are with the Faculty of Eng. and IT, Uni. of Tech., Sydney, Australia. E-mail: {jinjun.chen, xyzhanggz, chiyangit}@gmail.com.*
- *L.T. Yang is with the School of Comput. Sci. and Tech., Huazhong Uni. of Sci. and Tech., China, and also with the Dept. of Comput. Sci., St. Francis Xavier Uni., Canada. E-mail: ltyang@stfx.ca.*
- *R. Ranjan is with CSIRO Computational Informatics Division, Australia. E-mail: rranjans@gmail.com.*
- *R. Kotagiri is with the Dept. of Comput. and Information Systems, The Uni. of Melbourne, Australia. E-mail: kotagiri@unimelb.edu.au.*

Manuscript received 21 May 2013; revised 23 July 2013; accepted 24 July 2013. Date of publication 4 Aug. 2013; date of current version 13 Aug. 2014. Recommended for acceptance by Y. Xiang.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TPDS.2013.191

users who have high-level security demands over their data.

Although existing data auditing schemes already have various properties (see Section 2), potential risks and inefficiency such as security risks in unauthorized auditing requests and inefficiency in processing small updates still exist. In this paper, we will focus on better support for small dynamic updates, which benefits the scalability and efficiency of a cloud storage server. To achieve this, our scheme utilizes a flexible data segmentation strategy and a ranked Merkle hash tree (RMHT). Meanwhile, we will address a potential security problem in supporting public verifiability to make the scheme more secure and robust, which is achieved by adding an additional authorization process among the three participating parties of client, CSS and a third-party auditor (TPA).

Research contributions of this paper can be summarized as follows:

1. For the first time, we formally analyze different types of fine-grained dynamic data update requests on variable-sized file blocks in a single dataset. To the best of our knowledge, we are the first to propose a public auditing scheme based on BLS signature and Merkle hash tree (MHT) that can support fine-grained update requests. Compared to existing schemes, our scheme supports updates with a size that is not restricted by the size of file blocks, thereby offers extra flexibility and scalability compared to existing schemes.
2. For better security, our scheme incorporates an additional authorization process with the aim of eliminating threats of unauthorized audit challenges from malicious or pretended third-party auditors, which we term as 'authorized auditing'.
3. We investigate how to improve the efficiency in verifying frequent small updates which exist in many popular cloud and big data contexts such as social media. Accordingly, we propose a further enhancement for our scheme to make it more suitable for this situation than existing schemes. Compared to existing schemes, both theoretical analysis and experimental results demonstrate that our modified scheme can significantly lower communication overheads.

For the convenience of the readers, we list some frequently-used acronyms in Appendix 1 which is available in the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPDS.2013.191.

1.1 Paper Organization

The rest of this paper is organized as follows. Section 2 discusses related work. Section 3 provides motivating examples as well as a detailed analysis of our research problem. Section 4 provides a description of our proposed scheme in detail, with also a detailed analysis of fine-grained update requests and how they can be supported. Section 5 provides security analysis for our design. Section 6 provides experimental results. Section 7 concludes our research and points out future work.

2 Related Work

Compared to traditional systems, scalability and elasticity are key advantages of cloud [1], [2], [3]. As such, efficiency in supporting dynamic data is of great importance. Security and privacy protection on dynamic data has been studied extensively in the past [6], [8], [12], [17]. In this paper, we will focus on small and frequent data updates, which is important because these updates exist in many cloud applications such as business transactions and online social networks (e.g. Twitter [18]). Cloud users may also need to split big datasets into smaller datasets and store them in different physical servers for reliability, privacy-preserving or efficient processing purposes.

Among the most pressing problems related to cloud is data security/privacy [4], [5], [19]. It has been one of the most frequently raised concerns [5], [20]. There is a lot of work trying to enhance cloud data security/privacy with technological approaches on CSP side, such as [21], [22]. As discussed in Section 1, they are of equal importance as our focus of external verifications.

Integrity verification for outsourced data storage has attracted extensive research interest. The concept of proofs of retrievability (POR) and its first model was proposed by Jules *et al.* [14]. Unfortunately, their scheme can only be applied to static data storage such as archive or library. In the same year, Ateniese, *et al.* proposed a similar model named 'provable data possession' (PDP) [10]. Their schemes offer 'blockless verification' which means the verifier can verify the integrity of a proportion of the outsourced file through verifying a combination of pre-computed file tags which they call homomorphic verifiable tags (HVTs) or homomorphic linear authenticators (HLAs). Work by Shacham, *et al.* [15] provided an improved POR model with stateless verification. They also proposed a MAC-based private verification scheme and the first public verification scheme in the literature that based on BLS signature scheme [23]. In their second scheme, the generation and verification of integrity proofs are similar to signing and verification of BLS signatures. When wielding the same security strength (say, 80-bit security), a BLS signature (160 bit) is much shorter than an RSA signature (1024 bit), which is a desired benefit for a POR scheme. They also proved the security of both their schemes and the PDP scheme by Ateniese, *et al.* [9], [10]. From then on, the concepts of PDP and POR were in fact unified under this new compact POR model. Ateniese, *et al.* extended their scheme for enhanced scalability [8], but only partial data dynamics and a predefined number of challenges is supported. In 2009, Erway, *et al.* proposed the first PDP scheme based on skip list that can support full dynamic data updates [12]. However, public auditability and variable-sized file blocks are not supported by default. Wang, *et al.* [6] proposed a scheme based on BLS signature that can support public auditing (especially from a third-party auditor, TPA) and full data dynamics, which is one of the latest works on public data auditing with dynamics support. However, their scheme lacks support for fine-grained update and authorized auditing which are the main focuses of our work. Latest work by Wang *et al.* [7] added a random masking technology on top of [6] to ensure the TPA cannot infer the raw data file from a series of integrity

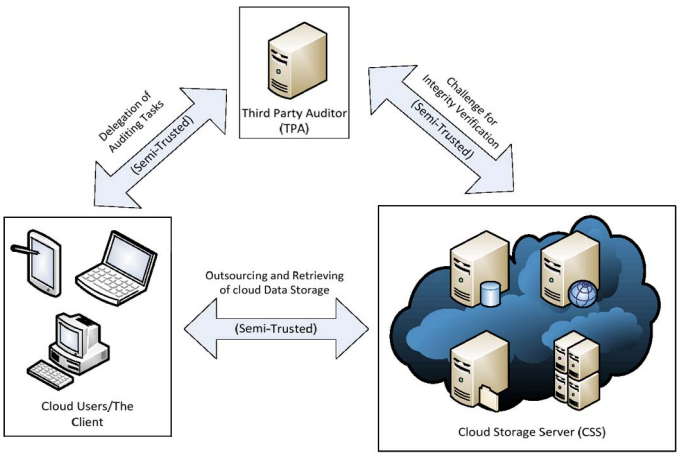


Fig. 1. Relationship between the participating parties in a public auditing scheme.

proofs. In their scheme, they also incorporated a strategy first proposed in [15] to segment file blocks into multiple 'sectors'. However, the use of this strategy was limited to trading-off storage cost with communication cost.

Other lines of research in this area include the work of Ateniese, *et al.* [24] on how to transform a mutual identification protocol to a PDP scheme; scheme by Zhu, *et al.* [13] that allows different service providers in a hybrid cloud to cooperatively prove data integrity to data owner; and the MR-PDP Scheme based on PDP [10] proposed by Curtmola, *et al.* [11] that can efficiently prove the integrity of multiple replicas along with the original data file.

3 Motivating Examples and Problem Analysis

3.1 Motivating Examples

Cost-efficiency brought by elasticity is one of the most important reasons why cloud is being widely adopted. For example, Vodafone Australia is currently using Amazon cloud to provide their users with mobile online-video-watching services. According to their statistics, the number of video requests per second (RPS) can reach an average of over 700 during less than 10 percent of the time such as Friday nights and public holidays, compared to a mere 70 in average in the rest 90 percent of the time. The variation in demand is more than 9 times [3]. Without cloud computing, Vodafone cannot avoid purchasing computing facilities that can process 700 RPS, but it will be a total waste for most of the time. This is where cloud computing can save a significant amount of investments—cloud's elasticity allows the user-purchased computation capacity to scale up or down on-the-fly at any time. Therefore, user requests can be fulfilled without wasting investments in computational powers. Other 2 large companies who own *news.com.au* and *realestate.com.au*, respectively, are using Amazon cloud for the same reason [3]. We can see through these cases that scalability and elasticity, thereby the capability and efficiency in supporting data dynamics, are of extreme importance in cloud computing.

Many big data applications will keep user data stored on the cloud for small-sized but very frequent updates. A most typical example is Twitter, where each tweet is restricted to 140 characters long (which equals 140 bytes in ASCII code). They can add up to a total of 12 terabytes of data per day [18]. Storage of transaction records in banking or securities markets is a similar and more security-heavy example. Moreover, cloud users may need to split large-scale datasets into smaller chunks before uploading to the cloud for privacy-preserving [17] or efficient scheduling [19]. In this regard, efficiency in processing small updates is always essential in big data applications.

To better support scalability and elasticity of cloud computing, some recent public data auditing schemes do support data dynamics. However, types of updates supported are limited. Therefore previous schemes may not be suitable for some practical scenarios. Besides, there is a potential security threat in the existing schemes. We will discuss these problems in detail in the next Section 3.2.

3.2 Problem Analysis

3.2.1 Roles of the Participating Parties

Most PDP and POR schemes can support public data verification. In such schemes, there are three participating parties: client, CSS and TPA. Relationships between the three parties are shown in Fig. 1. In brief, both CSS and TPA are only semi-trusted to the client. In the old model, the challenge message is very simple so that everyone can send a challenge to CSS for the proof of a certain set of file blocks, which can enable malicious exploits in practice. First, a malicious party can launch distributed denial-of-service (DDOS) attacks by sending multiple challenges from multiple clients at a time to cause additional overhead on CSS and congestion to its network connections, thereby causing degeneration of service qualities. Second, an adversary may get privacy-sensitive information from the integrity proofs returned by CSS. By challenging the CSS multiple times, an adversary can either get considerable information about user data (due to the fact that returned integrity proofs are computed with client-selected data blocks), or gather statistical information about cloud service status. To this end, traditional PDP models cannot quite meet the security requirements of 'auditing-as-a-service', even though they support public verifiability.

3.2.2 Verifiable Fine-Grained Dynamic Data Operations

Some of the existing public auditing schemes can already support full data dynamics [6], [7], [12]. In their models, only insertions, deletions and modifications on fixed-sized blocks are discussed. Particularly, in BLS-signature-based schemes [6], [7], [13], [15] with 80-bit security, size of each data block is either restricted by the 160-bit prime group order p, as each block is segmented into a fixed number of 160-bit sectors. This design is inherently unsuitable to support variable-sized blocks, despite their remarkable advantage of shorter integrity proofs. In fact, as described in Section 2, existing schemes can only support insertion, deletion or modification of one or multiple fixed-sized blocks, which we call 'coarse-grained' updates.

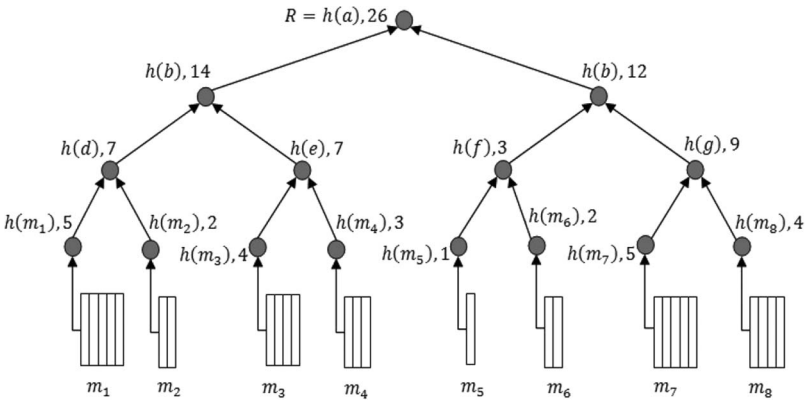


Fig. 2. Example of a rank-based Merkle hash tree (RMHT).

Although support for coarse-grained updates can provide an integrity verification scheme with basic scalability, data updating operations in practice can always be more complicated. For example, the verifiable update process introduced in [6], [12] cannot handle deletions or modifications in a size lesser than a block. For insertions, there is a simple extension that enables insertion of an arbitrary-sized dataset—CSS can always create a new block (or several blocks) for every insertion. However, when there are a large number of small upgrades (especially insertions), the amount of wasted storage will be huge. For example, in [6], [12] the recommended size for a data block is 16k bytes. For each insertion of a 140-byte Twitter message, more than 99 percent of the newly allocated storage is wasted—they cannot be reused until the block is deleted. These problems can all be resolved if fine-grained data updates are supported. According to this observation, supporting of fine-grained updates can bring not only additional flexibility, but also improved efficiency.

Our model assumes the following:

Assumption 1. *CSS will honestly answer all data queries to its clients. In other words, if a user asks to retrieve a certain piece of her data stored on CSS, CSS will not try to cheat her with an incorrect answer.*

This assumption—reliability—should be a basic service quality guarantee for cloud storage services.

PDP and POR are different models with similar goals. One main difference is that the file is encoded with error-correction code in the POR model, but not in the PDP model [6]. As in [6], [7], we will not restrict our work to either of the models.

4 The Proposed Scheme

Some common notations are introduced in Appendix A.

4.1 Preliminaries

4.1.1 Bilinear Map

Assume a group G is a gap Diffie-Hellman (GDH) group with prime order p. A bilinear map is a map constructed as $e : G \times G \rightarrow G_T$ where G_T is a multiplicative cyclic group with prime order. A useful e should have the following properties: bilinearity—$\forall m, n \in G \Rightarrow e(m^a, n^b) = e(m, n)^{ab}$; non-degeneracy—$\forall m \in G,\ m \neq 0 \Rightarrow e(m, m) \neq 1$; and computability—$e$ should be efficiently computable. For simplicity, we will use this symmetric bilinear map in our scheme description. Alternatively, the more efficient asymmetric bilinear map $e : G_1 \times G_2 \rightarrow G_T$ may also be applied, as was pointed out in [23].

4.1.2 Ranked Merkle Hash Tree (RMHT)

The Merkle Hash Tree (MHT) [25] has been intensively studied in the past. In this paper we utilize an extended MHT with ranks which we named RMHT. Similar to a binary tree, each node N will have a maximum of 2 child nodes. In fact, according to the update algorithm, every non-leaf node will constantly have 2 child nodes. Information contained in one node N in an RMHT T is represented as $\{\mathcal{H}, r_N\}$ where $\mathcal{H}$ is a hash value and r_N is the rank of this node. T is constructed as follows. For a leaf node LN based on a message m_i, we have $\mathcal{H} = h(m_i)$, $r_{LN} = s_i$; A parent node of $N_1 = \{\mathcal{H}_1, r_{N1}\}$ and $N_2 = \{\mathcal{H}_2, r_{N2}\}$ is constructed as $N_P = \{h(\mathcal{H}_1 \| \mathcal{H}_2), (r_{N1} + r_{N2})\}$ where $\|$ is a concatenation operator. A leaf node m_i's AAI Ω_i is a set of hash values chosen from every of its upper level so that the root value R can be computed through $\{m_i, \Omega_i\}$. For example, for the RMHT demonstrated in Fig. 2, m_1's AAI $\Omega_1 = \{h(m_2), h(e), h(d)\}$. According to the property of RMHT, we know that the number of hash values included in Ω_i equals the depth of m_i in T.

4.2 Framework and Definitions

We first define the following block-level fine-grained update operations:

Definition 1 (Types of Block-Level Operations in Fine-Grained Updates). *Block-level operations in fine-grained dynamic data updates may contain the following 6 types of operations: partial modification $\mathcal{PM}$—a consecutive part of a certain block needs to be updated; whole-block modification $\mathcal{M}$—a whole block needs to be replaced by a new set of data; block deletion $\mathcal{D}$—a whole block needs to be deleted from the tree structure; block insertion $\mathcal{J}$—a whole block needs to be created on the tree structure to contain newly inserted data;*

and block splitting $\mathcal{SP}$—a part of data in a block needs to be taken out to form a new block to be inserted next to it.[1]

Framework of public auditing scheme with data dynamics support is consisted of a series of algorithms. Similar to [12], the algorithms in our framework are: $Keygen$, $FilePreProc$, $Challenge$, $Verify$, $Genproof$, $PerformUpdate$ and $VerifyUpdate$. Detailed definitions and descriptions can be found in Appendix B.

4.3 Our Scheme

We now describe our proposed scheme in the aim of supporting variable-sized data blocks, authorized third-party auditing and fine-grained dynamic data updates.

4.3.1 Overview

Our scheme is described in three parts:

1. Setup: the client will generate keying materials via $KeyGen$ and $FileProc$, then upload the data to CSS. Different from previous schemes, the client will store a RMHT instead of a MHT as metadata. Moreover, the client will authorize the TPA by sharing a value sig_{AUTH}.
2. Verifiable Data Updating: the CSS performs the client's fine-grained update requests via $PerformUpdate$, then the client runs $VerifyUpdate$ to check whether CSS has performed the updates on both the data blocks and their corresponding authenticators (used for auditing) honestly.
3. Challenge, Proof Generation and Verification: Describes how the integrity of the data stored on CSS is verified by TPA via $GenChallenge$, $GenProof$ and $Verify$.

We now describe our scheme in detail as follows.

4.3.2 Setup

This phase is similar to the existing BLS-based schemes except for the segmentation of file blocks. Let $e : G \times G \rightarrow G_T$ be a bilinear map defined in Section 4.1, where G is a GDH group supported by $\mathbb{Z}_p$[2].$H : (0,1)^* \rightarrow G$ is a collision-resistant hash function, and h is another cryptographic hash function.

After all parties have finished negotiating the fundamental parameters above, the client runs the following algorithms:

$KeyGen(1^k)$: The client generates a secret value $\alpha \in \mathbb{Z}_p$ and a generator g of G, then compute $v = g^\alpha$. A secret signing key pair $\{spk, ssk\}$ is chosen with respect to a designated provably secure signature scheme whose signing algorithm is denoted as $\mathrm{Sig}()$. This algorithm outputs $\{ssk, \alpha\}$ as the secret key sk and $\{spk, v, g\}$ as the public key pk. For simplicity, in our settings, we use the same key pair for signatures, i.e., $ssk = \alpha, spk = \{v, g\}$.

$FilePreProc(F, sk, SegReq)$: According to the preemptively determined segmentation requirement $SegReq$ (including $s_{\max}$, a predefined upper-bound of the number of segments per block), segments file F into $\mathcal{F} = \{m_{ij}\}$, $i \in [1, l], j \in [1, s], s_i \in [1, s_{\max}]$, i.e., F is segmented into a total of l blocks, with the ith block having s_i segments. In our settings, every file segment should of the same size $\eta \in (0, p)$ and as large as possible (see [15]). Since $|p| = 20$ bytes is used in a BLS signature with 80-bit security (sufficient in practice), $\eta = 20$ bytes is a common choice. According to $s_{\max}$, a set $U = \{u_k \in \mathbb{Z}_p\} k \in [1, s_{\max}]$ is chosen so that the client can compute the HLAs σ_i for each block: $\sigma_i = (H(m_i) \prod_{j=1}^{s_i} u_j^{m_{ij}})^\alpha$ which constitute the ordered set $\Phi = \{\sigma_i\}_{i \in [1,l]}$. This is similar to signing a message with BLS signature. The client also generate a root R based on construction of an RMHT T over $H(m_i)$ and compute $sig = (H(R))^\alpha$. Finally, let $u = (u_1 \| \ldots \| u_{s_{\max}})$, the client compute the file tag for F as $t = name \| n \| u \| \mathrm{Sig}_{ssk}\ (name \| n \| u)$ and then output $\{\mathcal{F}, \Phi, T, R, sig, t\}$.

4.3.3 Prepare for Authorization

The client asks (her choice of) TPA for its ID *VID* (for security, *VID* is used for authorization only). TPA will then return its ID, encrypted with the client's public key. The client will then compute $sig_{AUTH} = \mathrm{Sig}_{ssk}(AUTH \| t \| VID)$ and sends sig_{AUTH} along with the auditing delegation request to TPA for it to compose a challenge later on.

Different from existing schemes, after the execution of the above two algorithms, the client will keep the RMHT 'skeleton' with only ranks of each node and indices of each file block to reduce fine-grained update requests to block-level operations. We will show how this can be done in Section 4.4. The client then sends $\{\mathcal{F}, t, \Phi, sig, AUTH\}$ to CSS and deletes $\{F, \mathcal{F}, t, \Phi, sig\}$ from its local storage. The CSS will construct an RMHT T based on m_i and keep T stored with $\{\mathcal{F}, t, \Phi, sig, AUTH\}$ for later verification, which should be identical to the tree spawned at client-side just a moment ago.

4.3.4 Verifiable Data Updating

Same as *Setup*, this process will also be between client and CSS. We discuss 5 types of block-level updates (operations) that will affect T: $\mathcal{PM}$, $\mathcal{M}$, $\mathcal{D}$, $\mathcal{J}$ and $\mathcal{SP}$ (see *Definition 1*). We will discuss how these requests can form fine-grained update requests in general in Section 4.4.

The verifiable data update process for a $\mathcal{PM}$-typed update is as follows (see Fig. 3):

1. The client composes an update quest $UpdateReq$ defined in Section 4.2 and sends it to CSS.

2. CSS executes the following algorithm:

$PerformUpdate(UpdateReq, \mathcal{F})$: CSS parses $UpdateReq$ and get $\{\mathcal{PM}, i, o, m_{new}\}$. When $Type = \mathcal{PM}$, CSS will update m_i and T accordingly, then output $P_{update} = \{m_i, \Omega_i, R', sig\}$ (note that Ω_i stays the same during the update) and the updated file $\mathcal{F}'$.

Upon finishing of this algorithm, CSS will send P_{update} to the client.

3. After receiving P_{update}, the client executes the following algorithm:

$VerifyUpdate(pk, P_{update})$: The client computes m_i' using $\{m_i, UpdateReq\}$, then parse P_{update} to $\{m_i, \Omega_i, R', sig\}$,

1. There are other possible operations such as block merging $\mathcal{ME}$—two blocks need to be merged into the first block before the second block is deleted, and data moving $\mathcal{MV}$—move a part of data from one block to another, if the size of the second block does not exceed $s_{\max} \cdot \eta$ after this update. However, the fine-grained update requests discussed in this paper do not involve these operations, thus we will omit them in our current discussion. We will leave the problem of how to exploit them in future work.

2. Most exponential operations in this paper are modulo p. Therefore, for simplicity, we will use g^σ instead of $g^\sigma \bmod p$ unless otherwise specified.

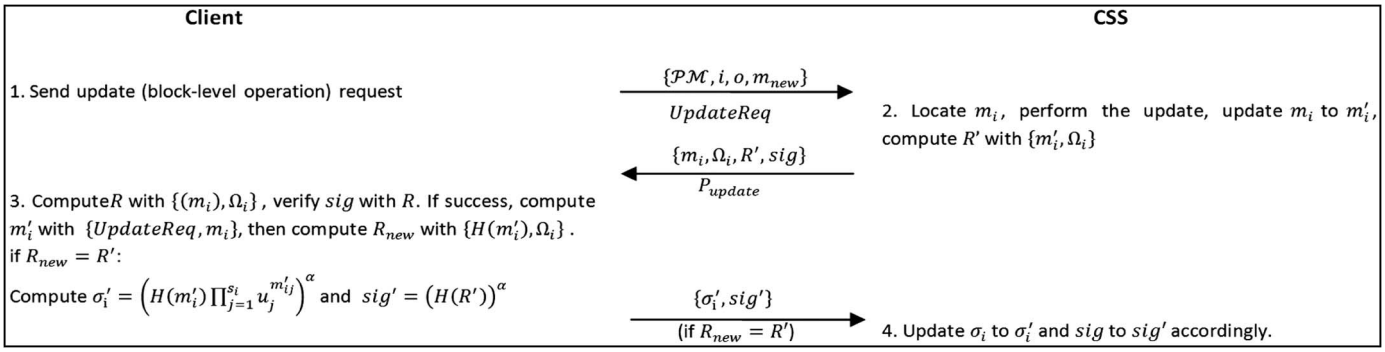


Fig. 3. Verifiable PM-typed Data Update in our scheme.

compute R (and $H(R)$) and R_{new} use $\{m_i, \Omega_i\}$ and $\{m_i', \Omega_i\}$ respectively. It verifies sig use $H(R)$, and check if $R_{new} = R'$. If either of these two verifications fails, then output $FALSE$ and return to CSS, otherwise output $TRUE$.

If the output of the algorithms is $TRUE$, then the client computes $\sigma_i' = (H(m_i')\prod_{j=1}^{S_1} u_j^{m_{ij}'})^{\alpha}$ and $sig' = (H(R'))^{\alpha}$ then sends $\{\sigma_i', sig'\}$ to CSS.

4. The CSS will update σ_i to σ_i' and sig to sig' accordingly and delete $\mathcal{F}$ if it receives $\{\sigma_i', sig'\}$, or it will run $PerformUpdate()$ again if it receives $FALSE$. A cheating CSS will fail the verification and constantly receive $FASLE$ until it performed the update as the client requested.

Due to their similarity to the process described above, other types of operations are only briefly discussed as follows. For whole-block operations $\mathcal{M}$, $\mathcal{D}$, and $\mathcal{J}$, as in model in the existing work [6], the client can directly compute σ_i' without retrieving data from the original file F stored on CSS, thus the client can send σ_i' along with $UpdateReq$ in the first phase. For responding to an update request, CSS only needs to send back $H(m_i)$ instead of m_i. Other operations will be similar to where $Type = \mathcal{PM}$. For an $\mathcal{SP}$-typed update, in addition to updating m_i to m_i', a new block m^* needs to be inserted to T after m_i'. Nonetheless, as the contents in m^* is a part of the old m_i, the CSS still needs to send m_i back to the client. The process afterwards will be just similar to a $\mathcal{PM}$-typed upgrade, with an only exception that the client will compute R_{new} using $\{m_i', h(m^*), \Omega_i\}$ to compare to R', instead of using $\{m_i', \Omega_i\}$ as in the $\mathcal{PM}$-typed update.

4.3.5 Challenge, Proof Generation and Verification

In our setting, TPA must show CSS that it is indeed authorized by the file owner before it can challenge a certain file.

1. TPA runs the following algorithm:

$GenChallenge(Acc, pk, sig_{AUTH})$: According to the accuracy required in this auditing, TPA will decide to verify c out of the total l blocks. Then, a challenge message $chal = \{sig_{AUTH}, \{VID\}_{PK_{CSS}}, \{i, v_i\}_{i\in I}\}$ is generated where VID is TPA's ID, I is a randomly selected subset of $[1, l]$ with c elements and $\{v_i \in \mathbb{Z}_p\}_{i\in I}$ are c randomly-chosen coefficients. Note that VID is encrypted with the CSS's public key PK_{CSS} so that CSS can later decrypt $\{VID\}_{PK_{CSS}}$ with the corresponding secret key.

TPA then sends $chal$ to CSS.

2. After receiving $chal$, CSS will run the following algorithm:

$GenProof(pk, F, sig_{AUTH}, \Phi, chal)$: Let $w = \max\{s_i\}_{i\in I}$. CSS will first verify sig_{AUTH} with $AUTH$, t, VID and the client's public key spk, and output $REJECT$ if it fails. Otherwise, CSS will compute $\mu_k = \sum_{i\in I} v_i m_{ik}, k \in [1, w]$ and $\sigma = \prod_{i\in I} \sigma_i^{v_i}$ and compose the proof P as $= \{\{\mu_k\}_{k\in[1,w]}, \{H(m_i), \Omega_i\}_{i\in I}, sig\}$, then ouput P. Note that during the computation of μ_k, we will let $m_{ik} = 0$ if $k > s_i$.

After execution of this algorithm, CSS will send P to TPA.

3. After receiving P, TPA will run the following algorithm:

$Verify(pk, chal, P)$: TPA will compute R using $\{H(m_i), \Omega_i\}$ and then verify sig use public keys g and v by comparing $e(sig, g)$ with $(H(R), v)$. If they are equal, let $\omega = \prod_{i\in I} H(m_i)^{v_i} \cdot \prod_{k\in[1,w]} u_k^{\mu_k}$, TPA will further check if $e(\sigma, g)$ equals $e(\omega, v)$, which is similar to verifying a BLS signature. If all the two equations hold then the algorithm returns $TRUE$, otherwise it returns $FALSE$.

An illustration of Challenge and Verification processes can be found in Fig. 4.

4.4 Analysis on Fine-Grained Dynamic Data Updates

Following the settings in our proposed scheme, we now define a fine-grained update request for an outsourced file divided into l variable-sized blocks, where each block is consisted of $s_i \in [1, s_{\max}]$ segments of a fixed size η each. Assume an RMHT T is built upon $\{m_i\}_{i\in[1,l]}$ for authentication, which means T must keep updated with each RMHT operation for CSS to send back the root R for the client to verify the correctness of this operation (see Section 4.3). We now try to define and categorize all types of fine-grained updates, and then analyze the RMHT operations with $Type = \mathcal{PM}, \mathcal{M}, \mathcal{D}, \mathcal{J}$ or $\mathcal{SP}$ that will be invoked along with the update of the data file.

Definition 2 (Fine-Grained Data Update Request). *A fine-grained update request is defined as $FReq = \{o, len, m_{new}\}$, where o indicates the starting offset of this update in F, len indicates the data length after o that needs to be updated (so that $\{o, len\}$ can characterize an exact proportion of the original file F that needs to be updated, which we will later call m_{old}), and m_{new} is the new message to be inserted into F from offset o.*

We assume the data needed to be obsolete and the new data to be added shares a common starting offset o in F, as

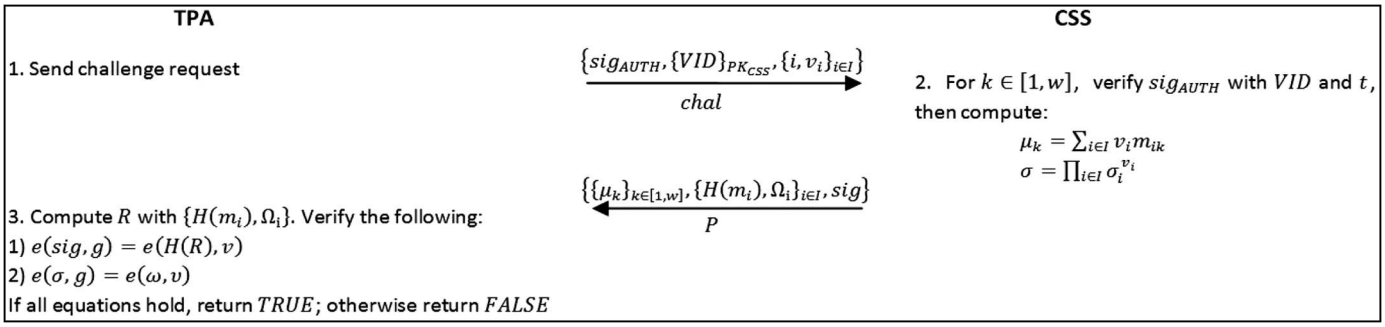


Fig. 4. Challenge, Proof Generation and Verification in our scheme.

otherwise it can be split into multiple updates defined in *Definition 2* commencing in sequence. We now introduce a rule to follow during all update processes:

Condition 1 (Block Size Limits in Updates). *An update operation must not cause the size of any block to exceed $s_{\max}$; After any operation, a block that has 0 bit data remaining must be deleted from T.*

Detailed analysis can be found in Appendix C, which can be summarized as the following theorem:

Theorem 1. *Any valid fine-grained update request that is in the form of $\{o, len, m_{new}\}$ can either directly belong to, or be split into some smaller requests that belong to, the following 5 types of block-level update requests: $\mathcal{PM}$, $\mathcal{M}$, $\mathcal{J}$, $\mathcal{D}$ and $\mathcal{SP}$.*

Proof. See Appendix C. □

Through the analysis above, we know that a large number of small updates, no matter insert, delete or modification, will always invoke a large number of $\mathcal{PM}$ operations. We now try to optimize $\mathcal{PM}$ operations in the next section to make it more efficient.

4.5 Modification for Better Support of Small Updates

Although our proposed scheme can support fine-grained update requests, the client still needs to retrieve the entire file block from CSS to compute the new HLA, in the sense that the client is the only party that has the secret key α to compute the new HLA but clients do not have F stored locally. Therefore, the additional cost in communication will be huge for frequent updates. In this section, we will propose a modification to address this problem, utilizing the fact that CSS only needs to send back data in the block that stayed unchanged.

The framework we use here is identical to the one used in our scheme introduced in Section 4.2 (which we will also name as 'the basic scheme' hereafter). Changes are made in $PerformUpdate$ and $VerifyUpdate$; Setup, Challenge, Proof Generation and Verification phases are same as in our basic scheme. Therefore, we will only describe the two algorithms in the following phase:

4.5.1 Verifiable Data Updating

We also discuss $\mathcal{PM}$ operations here first.

$PerformUpdate$: After CSS has received the update request $UpdateReq$ from the client, it will parse it as $\{\mathcal{PM}, I, o, m_{new}\}$ and use $\{o, |m_{new}|\}$ to gather the sectors that are not involved in this update, which we denote as $\{m_{ij}\}_{j\in M}$. CSS will then perform the update to get m_i', then compute R', then send the proof of update $P_{update} = \{\{m_{ij}\}_{j\in M}, H(m_i), \Omega_i, R', sig\}$ to the client.

$VerifyUpdate$: After the client received $H(m_i)$, it will first compute R using $H(m_i), \Omega_i$ and verify sig, then it will compute m_i' using $\{\{m_{ij}\}_{j\in M}, m_{new}\}$ and then compute R_{new} with $\{m_i', \Omega_i\}$ and compare R_{new} with R'. If $R_{new} = R'$, then the client will return $\{\sigma_i', sig'\}$ to CSS for it to update accordingly.

For an $\mathcal{SP}$ operation the process will be the same to our basic scheme as there are no new data inserted into T, therefore the retrieving of the entire data block is inevitable when computations of σ_i' and σ^* are required. For other types of operations, no old data is involved in new blocks; therefore the processes will also remain the same. The process is shown in Fig. 5.

4.6 Extensions and Generalizations

Our strategy can also be applied in RSA-based PDP or POR schemes to achieve authorized auditing and fine-grained data update requests. As RSA can inherently support variable-sized blocks, the process will be even easier. The batch auditing variation in [6], [7] can also be applied to our scheme, as we did not change the construction of HLAs and the verifications on them.

For the same reason, the random masking strategy for privacy preserving proposed in [7] can also be incorporated into our scheme to prevent TPA from parsing the challenged file blocks through a series of integrity proofs to a same set of blocks. Alternatively, we can also restrict the number of challenges to the same subset of data blocks. When data updates are frequent enough, the success rate of this attack will drop dramatically, because there is a high probability that one or many of the challenged blocks have already updated before c challenges are completed, which is the reason we did not incorporate this strategy into our scheme.

5 Security Analysis

In this section, the soundness and security of our scheme is discussed separately in phases, as the aim and behavior of the malicious adversary in each phase of our scheme is different.

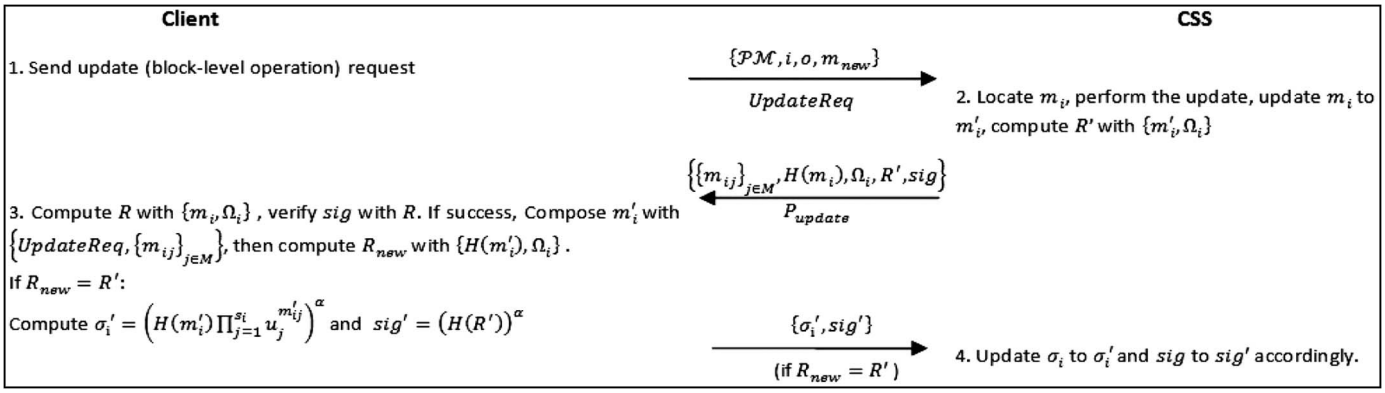


Fig. 5. Verifiable PM-typed Data Update in our modified scheme.

5.1 Challenge and Verification

In the challenge/verification process of our scheme, we try to secure the scheme against a malicious CSS who tries to cheat the verifier TPA about the integrity status of the client's data, which is the same as previous work on both PDP and POR. In this step, aside from the new authorization process (which will be discussed in detail later in this section), the only difference compared to [6] is the RMHT and variable-sectored blocks. Therefore, the security of this phase can be proven through a process highly similar with [6], using the same framework, adversarial model and interactive games defined in [6]. A detailed security proof for this phase is therefore omitted here.

5.2 TPA Authorization

Security of the new authorization strategy in our scheme is based on the existential unforgeability of the chosen signature scheme. We first define the behavior of a malicious third-party auditor.

Definition 3 (Malicious TPA). *A malicious TPA is a third party who aims at challenging a user's data stored on CSS for integrity proof without this user's permission. The malicious TPA has access to the entire network.*

According to this definition, none of the previous data auditing schemes is resilient against a malicious TPA. Now, in our scheme, we have the following theorem:

Theorem 2. *Through the authorization process, no malicious TPA can cause the CSS to respond with an integrity proof P over an arbitrary subset of file F, namely $m_i, i \in I$, unless a negligible probability.*

Proof. See Appendix D. □

From this theorem, we can see that the security of a public auditing scheme is strengthened by adding the authorization process. In fact, the scheme is now resilient against malicious or pretended auditing requests, as well as potential DDOS attacks launched by malicious auditors.

For even higher security, the client may mix in a nonce to the authorization message to make every auditing message distinct, so that no one can utilize a previous authorization message. However, this setting may not be appropriate for many scenarios, as the client must stay online when each auditing happens.

5.3 Verifiable Data Updating

In the verifiable updating process, the main adversary is the untrustworthy CSS who did not carry out the data update successfully, but still manages to return a satisfactory response to the client thereafter. We now illustrate the security of this phase of our scheme in the following theorem:

Theorem 3. *In the verifiable update process in both our basic scheme and the modification, CSS cannot provide the client with the satisfactory result, i.e., R' cannot match the R_{new} computed by the client with $\{H(m_i'), \Omega_i\}$, if CSS did not update the data as requested.*

Proof. See Appendix D. □

Note that in the verifiable update process, data retrieval is a part of the verifiable update process. According to *Assumption 1*, CSS will respond this query with the correct m_{ii}. If not with *Assumption 1*, it is recommended to independently retrieve $\{m_{ij}\}_{j\in M}$ before the update so that CSS cannot cheat the client intentionally, as it cannot distinguish whether the following update is based on this retrieval.

If CSS can be trusted even more, the client may let CSS compute $(u_j^{m_{ij}})^{\alpha}$ (where m_{ij} are the sectors that did not change) and send it back to the client, then the client will be able to compute σ' using it along with m_{new} and $H(m_i')$. This will keep the communication cost of this phase on a constantly low level. However, as the CSS is only considered semi-trusted and it is difficult for the client to verify $(\mu_j^{m_{ij}})^{\alpha}$ without m_{ij}, this assumption is unfortunately too strong for the majority of scenarios.

6 Evaluation and Experimental Results

We have provided an overall evaluation and comparison in Appendix E.

We conducted our experiments on U-Cloud—a cloud computing environment located in University of Technology, Sydney (UTS). The computing facilities of this system are located in several labs in the Faculty of Engineering and IT, UTS. On top of hardware and Linux OS, We installed KVM Hypervisor [26] which virtualizes the infrastructure and allows it to provide unified computing and storage resources. Upon virtualized data centers, Hadoop [27] is installed to facilitate the MapReduce programming model and distributed file system. Moreover, we installed OpenStack

open source cloud platform [28] which is responsible for global management, resource scheduling, task distribution and interaction with users.

We implemented both our scheme and its modification on U-Cloud, using a virtual machine with 36 CPU cores, 32GB RAM and 1TB storage in total. As in previous work [6], [12], we also used a 1GB randomly generated dataset for testing. The scheme is implemented under 80-bit security, i.e., $\eta = |p| = 160$ bits. As the number of sectors s (per block) is one of the most influential metrics to overall performance, we will use it as our primary metrics. For saving of the first wave of allocated storage, we used $s_i = s_{\max}$ in the initial data splitting and uploading. Note that $s_{\max}$ decides the total number of blocks for an arbitrary $|F|$. However, according to [10], the number of authenticated blocks is a constant with respect to a certain percentage of file tampered and a certain success rate of detection, therefore we will not take the number of audited blocks as our primary variable of measurement. All experimental results are an average of 20 runs.

We first tested how $s_{\max}$ can influence the size of proof P, which is missing in former schemes [6], [7]. From Fig. 6, we can see that generally the proof size decreases when $s_{\max}$ increases, because the average depth of leaf nodes m_i of T decreases when $s_{\max}$ increases to a certain level, especially when right after the initial uploading of $\mathcal{F}$. Note that the storage of HLA and RMHT at CSS side will also decrease with the increase of the average number of blocks. Therefore, a relatively large $s_{\max}$ (but not too large, which we will discuss along with the third experiment) is recommended in our dynamic setting.

Second, we tested the storage overhead for small insertions. Without support for fine-grained updates, every small insertion will cause creation of a whole new block and update of related MHT nodes, which is why our scheme has efficiency advantage. We compared our scheme against a representative (and also recent) public auditing scheme [6]. For comparison, we extended the older scheme a bit to let it support the communication-storage trade-off introduced in [15] so that it can support larger file blocks with multiple (but only a predefined constant number of) sectors each. The updates chosen for experiments are $10 * 140$ Bytes and $10 * 280$ Bytes, filled with random data. Results are shown in Figs. 7 and 8. For updates of the same total size, the increased storage on CSS for our scheme stays constant, while in the extended old scheme [6] (see Section 3.2.2) the storage increases linearly with the increase in size of the affected block. These results demonstrated that our scheme with fine-grained data update support can incur significantly lower storage overhead (down to $0.14\times$ in our test scenarios) for small insertions when compared to existing scheme.

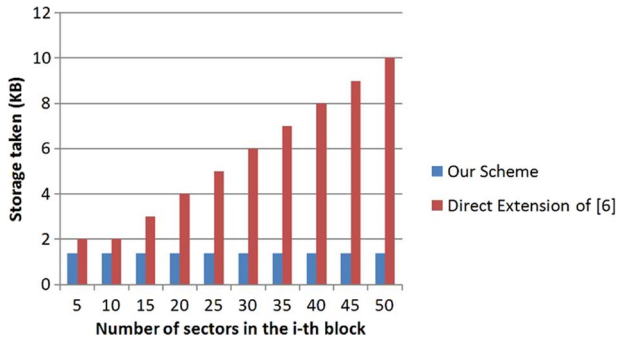


Fig. 7. Comparison of the total storage overhead invoked by $10*$ 140-byte insertions to the i-th block in our scheme, as opposed to the direct extension of [6].

Third, we investigated the performance improvement of the modification introduced in Section 4.5. We used 3 pieces of random data with sizes of 100 bytes, 140 bytes and 180 bytes, respectively, to update several blocks that contain 10 to 50 standard 20-byte sectors each. Data retrieval is a key factor of communication overheads in the verifiable update phase. For each update, we recorded the total amount of data retrieval for both our modified scheme and our basic scheme. The results in comparison are shown in Fig. 9. We can see that our modified scheme always has better efficiency with respect to data-retrieval-invoked communication overheads, and the advantage is more significant for larger updates. However, for an update of the same size, the advantage will decrease with the increase of $|s_i|$ where a larger number of sectors in the original file are needed to be retrieved. Therefore, the block size needs to be kept low if less communication in verifiable updates is highly demanded.

From the experimental results on small updates, we can see that our scheme can incur significantly lower storage overhead while our modified scheme can dramatically reduce communication overheads compared to the existing scheme. In practice, the important parameter $s_{\max}$ should

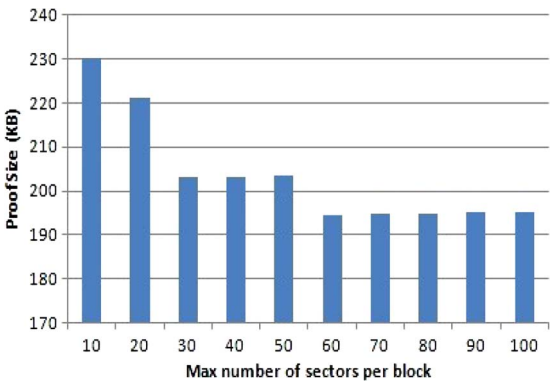


Fig. 6. Communication overhead invoked by an integrity proof with 80-bit security under different $s_{\max}$ for a 1GB data.

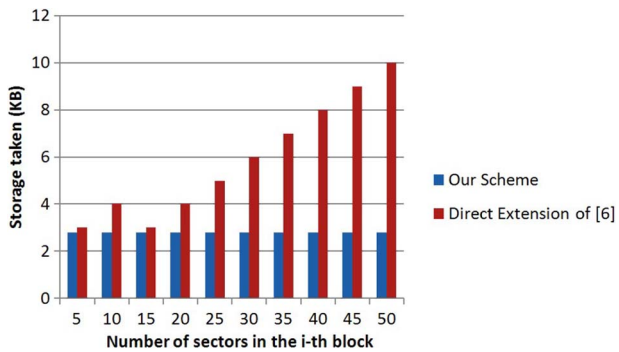


Fig. 8. Comparison of the total storage overhead invoked by $10*$ 280-byte insertions to the i-th block in our scheme, as opposed to the direct extension of [6].

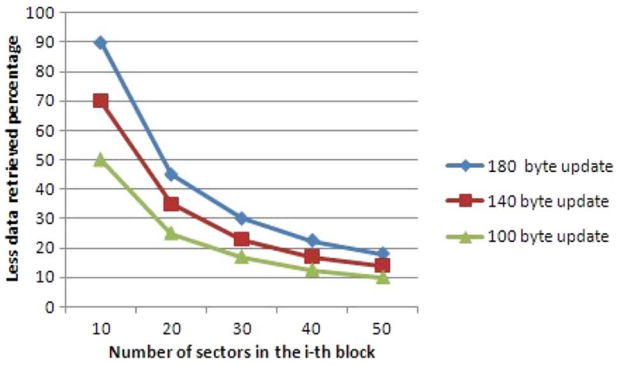


Fig. 9. Percentage in saving of communication overhead in data retrieval in the modified scheme, compared to our basic scheme.

be carefully chosen according to different data size and different efficiency demands in storage or communications. For example, for general applications with a similar scale (1GB per dataset and frequent 140-byte updates), a choice of $s_{\max} = 30$ will allow the scheme to incur significantly lowered overheads in both storage and communications during updates. Additional analysis regarding efficiency can be found in Appendix F.

7 Conclusion and Future Work

In this paper, we have provided a formal analysis on possible types of fine-grained data updates and proposed a scheme that can fully support authorized auditing and fine-grained update requests. Based on our scheme, we have also proposed a modification that can dramatically reduce communication overheads for verifications of small updates. Theoretical analysis and experimental results have demonstrated that our scheme can offer not only enhanced security and flexibility, but also significantly lower overheads for big data applications with a large number of frequent small updates such as applications in social media and business transactions.

Based on the contributions of this paper on improved data auditing, we plan to further investigate the next step on how to improve other server-side protection methods for efficient data security with effective data confidentiality and availability. Besides, we also plan to investigate auditability-aware data scheduling in cloud computing. As data security is also considered as a metric of quality-of-service (QoS) along with other metrics such as storage and computation, a highly efficient security-aware scheduling scheme will play an essential role under most cloud computing contexts.

Acknowledgment

This research work is partly supported by Australian Research Council under Linkage Project LP0990393.

References

[1] R. Buyya, C.S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, ''Cloud Computing and Emerging IT Platforms: Vision, Hype, Reality for Delivering Computing as the 5th Utility,'' *Future Gen. Comput. Syst.*, vol. 25, no. 6, pp. 599-616, June 2009.

[2] M. Armbrust, A. Fox, R. Griffith, A.D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, ''A View of Cloud Computing,'' *Commun. ACM*, vol. 53, no. 4, pp. 50-58, Apr. 2010.

[3] Customer Presentations on Amazon Summit Australia, Sydney, 2012, accessed on: March 25, 2013. [Online]. Available: http://aws.amazon.com/apac/awssummit-au/

[4] J. Yao, S. Chen, S. Nepal, D. Levy, and J. Zic, ''TrustStore: Making Amazon S3 Trustworthy With Services Composition,'' in *Proc. 10th IEEE/ACM Int'l Symposium on Cluster, Cloud and Grid Computing (CCGRID)*, 2010, pp. 600-605.

[5] D. Zissis and D. Lekkas, ''Addressing Cloud Computing Security Issues,'' *Future Gen. Comput. Syst.*, vol. 28, no. 3, pp. 583-592, Mar. 2011.

[6] Q. Wang, C. Wang, K. Ren, W. Lou, and J. Li, ''Enabling Public Auditability and Data Dynamics for Storage Security in Cloud Computing,'' *IEEE Trans. Parallel Distrib. Syst.*, vol. 22, no. 5, pp. 847-859, May 2011.

[7] C. Wang, Q. Wang, K. Ren, and W. Lou, ''Privacy-Preserving Public Auditing for Data Storage Security in Cloud Computing,'' in *Proc. 30st IEEE Conf. on Comput. and Commun. (INFOCOM)*, 2010, pp. 1-9.

[8] G. Ateniese, R.D. Pietro, L.V. Mancini, and G. Tsudik, ''Scalable and Efficient Provable Data Possession,'' in *Proc. 4th Int'l Conf. Security and Privacy in Commun. Netw. (SecureComm)*, 2008, pp. 1-10.

[9] G. Ateniese, R. Burns, R. Curtmola, J. Herring, O. Khan, L. Kissner, Z. Peterson, and D. Song, ''Remote Data Checking Using Provable Data Possession,'' *ACM Trans. Inf. Syst. Security*, vol. 14, no. 1, May 2011, Article 12.

[10] G. Ateniese, R.B. Johns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song, ''Provable Data Possession at Untrusted Stores,'' in *Proc. 14th ACM Conf. on Comput. and Commun. Security (CCS)*, 2007, pp. 598-609.

[11] R. Curtmola, O. Khan, R.C. Burns, and G. Ateniese, ''MR-PDP: Multiple-Replica Provable Data Possession,'' in *Proc. 28th IEEE Conf. on Distrib. Comput. Syst. (ICDCS)*, 2008, pp. 411-420.

[12] C. Erway, A. Küpçü, C. Papamanthou, and R. Tamassia, ''Dynamic Provable Data Possession,'' in *Proc. 16th ACM Conf. on Comput. and Commun. Security (CCS)*, 2009, pp. 213-222.

[13] Y. Zhu, H. Hu, G.-J. Ahn, and M. Yu, ''Cooperative Provable Data Possession for Integrity Verification in Multi-Cloud Storage,'' *IEEE Trans. Parallel Distrib. Syst.*, vol. 23, no. 12, pp. 2231-2244, Dec. 2012.

[14] A. Juels and B.S. Kaliski Jr., ''PORs: Proofs of Retrievability for Large Files,'' in *Proc. 14th ACM Conf. on Comput. and Commun. Security (CCS)*, 2007, pp. 584-597.

[15] H. Shacham and B. Waters, ''Compact Proofs of Retrievability,'' in *Proc. 14th Int'l Conf. on Theory and Appl. of Cryptol. and Inf. Security (ASIACRYPT)*, 2008, pp. 90-107.

[16] S. Nepal, S. Chen, J. Yao, and D. Thilakanathan, ''DIaaS: Data Integrity as a Service in the Cloud,'' in *Proc. 4th Int'l Conf. on Cloud Computing (IEEE CLOUD)*, 2011, pp. 308-315.

[17] Y. He, S. Barman, and J.F. Naughton, ''Preventing Equivalence Attacks in Updated, Anonymized Data,'' in *Proc. 27th IEEE Int'l Conf. on Data Engineering (ICDE)*, 2011, pp. 529-540.

[18] E. Naone, ''What Twitter Learns From All Those Tweets,'' in *Technology Review*, Sept. 2010, accessed on: March 25, 2013. [Online]. Available: http://www.technologyreview.com/view/420968/what-twitter-learns-from-all-those-tweets/

[19] X. Zhang, L.T. Yang, C. Liu, and J. Chen, ''A Scalable Two-Phase Top-Down Specialization Approach for Data Anonymization Using MapReduce on Cloud,'' *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 2, pp. 363-373, Feb. 2014.

[20] S.E. Schmidt, ''Security and Privacy in the AWS Cloud,'' presented at the Presentation Amazon Summit Australia, Sydney, Australia, May 2012, accessed on: March 25, 2013. [Online]. Available: http://aws.amazon.com/apac/awssummit-au/

[21] C. Liu, X. Zhang, C. Yang, and J. Chen, ''CCBKE—Session Key Negotiation for Fast and Secure Scheduling of Scientific Applications in Cloud Computing,'' *Future Gen. Comput. Syst.*, vol. 29, no. 5, pp. 1300-1308, July 2013.

[22] X. Zhang, C. Liu, S. Nepal, S. Panley, and J. Chen, ''A Privacy Leakage Upper-Bound Constraint Based Approach for Cost-Effective Privacy Preserving of Intermediate Datasets in Cloud,'' *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 6, pp. 1192-1202, June 2013.

Automated Privacy Audits to Complement the Notion of Control for Identity Management

Rafael Accorsi

Department of Telematics
Albert-Ludwigs-Universität Freiburg, Germany
accorsi@iig.uni-freiburg.de

Abstract. Identity management systems are indispensable in modern networked computing, as they equip data providers with key techniques to avoid the imminent privacy threats intrinsic to such environments. Their rationale is to convey data providers with a sense of *control* over the disclosure and usage of personal data to varying degree, so that they can take an active role in protecting their privacy. However, we purport the thesis that a holistic sense of control includes not only the *regulation* of disclosure, as identity management techniques currently do, but must equivalently comprise the *supervision* of compliance, i.e. credible evidence that data consumers behave according to the policies previously agreed upon. Despite its relevance, supervision has so far not been possible. We introduce the concept of *privacy evidence* and present the necessary technical building blocks to realise it in dynamic systems.

1 Introduction

In a technological setting where some even prophesy the death of privacy [5], the need for approaches to mediate and legislate for the collection of personal attributes and their usage is increasingly gaining in momentum and relevance. While such an investigation involves interdisciplinary efforts, we focus on the technical aspects. In this context, identity management systems (henceforth IMS) play an essential role in circumventing the privacy threats inherent to the deployment of information technology. They allow data providers to selectively disclose attributes to data consumers, possibly enabling data providers to formulate policies under which collected attributes can or cannot be employed.

The rationale of IMS is to convey a sense of control to data providers, where the "control" stands for the *regulation* of attribute disclosure. However, data providers today obtain no indication as to whether data consumers actually behave according to the policies agreed upon. Put other way, data providers are left with a number of privacy promises or expectations, but obtain no creditable evidence that their policies have been

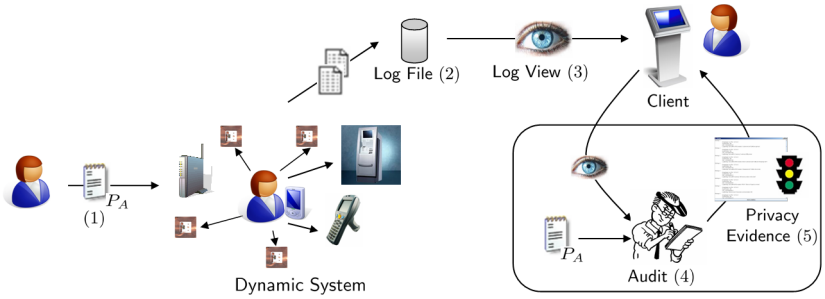


Fig. 1. The workflow for privacy evidence.

to realise the concept of privacy evidence. These building blocks are then described sequentially: in §3, we introduce a language for the expression of privacy policies; in §4, log views based on a secure logging service are presented; and in §5, we describe our approach to auditing log views based on the stated privacy policies. We discuss our work and provide perspectives for further work in §6.

2 Technical Setting and Building Blocks

The realisation of privacy evidence anticipates the steps depicted in Fig. 1. In (1), a data provider A formulates a policy P_A and communicates it to the data consumer. Since we consider dynamic systems with implicit interactions, we assume that policies are communicated before joining the system. (Implicit interactions take place without the awareness of the data provider.) When interacting with the system, a number of events are recorded as entries in log files (2). In fact, we assume that *every* event is recorded, so that log files offer a complete digital representation of the activity in a dynamic system. At some point in time the data consumer may retrieve the log view S_A containing all the log entries related to A (3). A can then visualise the collected data and start a third-party automated audit process (4) to check whether the policies P_A have been adhered to, thereby generating the corresponding privacy evidence (5).

To realise privacy evidence, the following central technical building blocks are essential: a *policy language* for the expression of privacy preferences in dynamic systems; *log views* to allow the visualisation of recording activity; a *secure logging* to ensure the authenticity of recorded data, in particular to improve the credibility of log views; and an *automated audit* process for checking the adherence to policies. In the forthcoming sections, we describe the work towards the realisation of privacy evidence.

Assumptions. In our work, we consider the following assumptions. First, *every* event happening in the system, as well as every access to collected data is recorded as an event in a log file. Second, on interacting with the system, data providers are identified while the events they are involved in are recorded. That is, the entries in the log file are always related to a data provider. Third, while the system is dynamic in that it adapts itself to the data providers' preferences, it is static regarding the data collection possibilities. Technically, this means that the ontology describing the system does not change over time and, hence, the policies of data providers do not become obsolete. Although these assumptions do not hold in general, they hold for some scenarios, as the one we consider in §6.

3 A Policy Language for Dynamic Systems

A policy language allows data providers to specify a set of rules, i.e. a policy to regulate the access to their attributes, whereas execution monitors on the data consumers' side enforce these rules and record the authorisation decisions for further inspection. However, in dynamic systems the sole expression of access rights is not enough. Policies for dynamic systems should also allow data providers to express which attributes may or may not be collected. The policy language we propose therefore builds on two notions: *access* and *collection*. In contexts where the distinction between these notions is irrelevant, we simply refer to them as an *act*.

We enrich atomic acts with conditions for *usage control*. Usage control extends traditional access control techniques by allowing data providers to specify *provisions* and *obligations* [11]. Intuitively, provisions express the conditions that must be fulfilled in order to grant or deny an act [7]. For example, access to the profile of data provider A is granted only for accounting purposes. Obligations express events that must occur once an act is granted or denied. For example, data provider A wants to be notified whenever the collection of attributes via RFID readers take place.

Figure 2 depicts the core definition of our policy language in BNF-notation. Intuitively, the policy of a data provider A is a finite set of rules $P_A = \{r_1, \ldots, r_n\}$ (Line 1), each of which can stand for a (conditional) act, i.e. collection or access regulation (Lines 2 and 3). When formulating a collection rule, A stipulates whether a certain subject is able to collect an attribute and/or event (Line 4). The same applies for the formulation of access rules (Line 5). In both cases, the wildcard $*$ can be used to represent a whole class of, e.g., subjects or attributes. Conditions can include provisions and obligations (Line 7): provisions regard the role a

```
 1.  <Policy>     := (<Rule>) | (<Rule>), <Policy>
 2.  <Rule>       := <Col_Ctrl> | <Col_Ctrl>, if (<Cond>) |
 3.                  <Acc_Ctrl> | <Acc_Ctrl>, if (<Cond>)
 4.  <Col_Ctrl>   := <Perm>, <Subj>, <Obj>, <Event>
 5.  <Acc_Ctrl>   := <Perm>, <Subj>, <Obj>, <Right>
 6.  <Cond>       := <Atom_Cond> | <Atom_Cond> && <Cond>
 7.  <Atom_Cond>  := <Provision> | <Obligation>
 8.  <Provision>  := role <Op> <Role> | purpose <Op> <Purpose>  |
 9.                  <DataField> <Op> <Value>
10.  <Obligation> := delete <DataField>  <Temp_mod> [<Sanction>] |
11.                  notify <DataProvider> <Temp_mod> [<Sanction>]
12.  <Perm>       := allow | deny
13.  <Right>      := read | write | exec <Cmd>
14.  <Temp_mod>   := immediately | within <Nat_Number> days
15.  <Sanction>   := otherwise <String>
16.  <Op>         := > | < | >= | <= | == | !=
```

Fig. 2. Policy language for dynamic systems.

subject takes, as well as the purpose of the access or collection and the value of collected data fields serving as guards (Lines 8 and 9); obligations encompass the deletion of some attribute within a certain timeframe and the notification of individuals (Lines 10 and 11). Obligations may or may well not include sanctions that hold in case a particular obligation is not fulfilled (Line 15).

The actual value of terminals, such as `Obj` and `Subj` are application-dependent and omitted here for simplicity. (To this end, we have defined data models corresponding to our scenario.) To illustrate how formulated rules appear and exemplify their expressive power, in Fig. 3 we consider two rules for the data provider A. Rule r_1, stipulates that A grants read access to his attributes provided the accessing subject adopts the role "Marketing", the purpose is personalised service and the accessed attribute is deleted within 30 days. In the case of non-adherence, a compensation of $100 is due. Rule r_2 prohibits the collection of data by RFID-readers.

4 Secure Logging and Log Views

Log data is a central source of information in computer systems. In contrast to other rather "static" files, such as text documents or spreadsheets, log files allow one to reconstruct the dynamics of a system, i.e. the course of events that led to some particular state. Hence, log files are a central

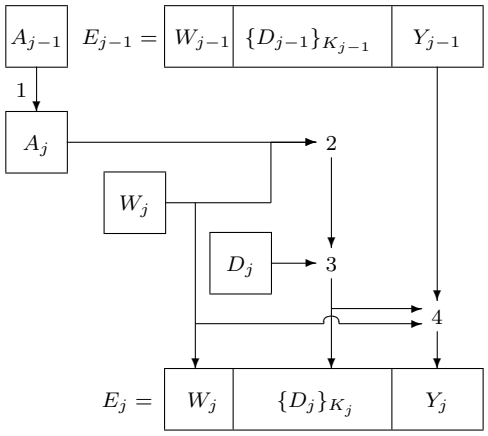


Fig. 4. Adding an entry to the log file.

encrypt log entries. Thus, we assume that the computation of the new value irretrievably overwrites the previous value.

2. $K_j = \mathit{Hash}(W_j, A_j)$ is the cryptographic key with which the jth log entry is encrypted. This key is based on the index W_j, so that only corresponding data providers gain access to the entry.
3. $\{D_j\}_{K_j}$ is the encrypted log entry D_j.
4. $Y_j = \mathit{Hash}(Y_{j-1}, \{D_j\}_{K_j}, W_j)$ is the jth value of the hash chain. Each link of the hash chain is based on the corresponding encrypted value of the log data.

The generated log entry, denoted $E_j = W_j, \{D_j\}_{K_j}, Y_j$, consists of the index W_j, the encrypted log entry $\{D_j\}_{K_j}$, and the hash chain value Y_j.

4.1 Log Views and their Generation

A central concept to allow supervision is to furnish data providers with timestamped information regarding *which* attributes have been collected, *who* has had access to them and *how* collected attributes have been used. In our approach, these pieces of information are compiled into a *log view* [12], a concept bearing similarity with its homonymous counterpart in the field of databases.

Log views are individualised audit trails consisting of factual data (performed transactions, collected attributes, etc.) and monitored data (access and usage information) about a particular data provider, as well as meta data – in the form of a digital signature – about the generating

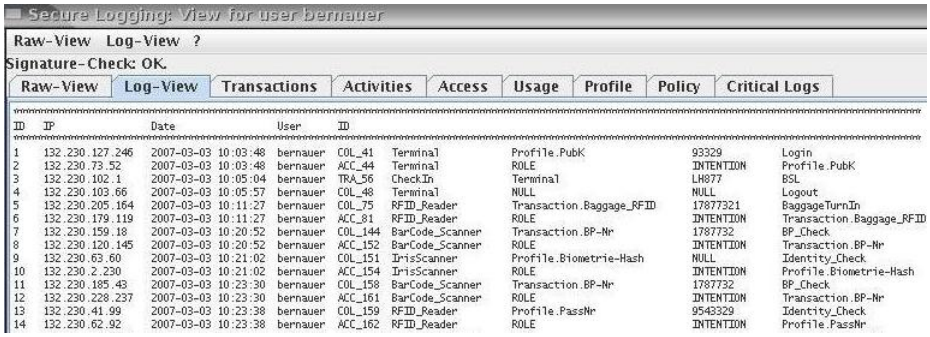


Fig. 5. Part of a log view for data provider `bernauer`.

data consumer and the integrity of a view. Figure 5 illustrates a part of log view of a data provider referred to as "bernauer".

As for the generation of log views, to retrieve a log view S_A the data provider A employs a trusted device (e.g. a home computer or a terminal dedicated to this purpose) to authenticate himself to the data consumer, who then starts a query over (possibly distributed) log files. Intuitively, the index of each entry is checked against the authenticated data provider. If they match and the entry passes an integrity check (based on the hash chain), then the content of the entry is decrypted and added to the log view of A. When all the entries are queried, the resultant view is signed and sent back to the inquiring data provider.

5 Automated Audits and Digital Privacy Evidence

Log views would, at least in theory, suffice to realise the holistic sense of control we argue for in this manuscript: data providers could browse through their log views and check whether their privacy policies have been adhered to or not. However, this is more intricate than it seems. Log views can easily include thousands of entries and their interrelationships are often hard to comprehend and reconstruct, regardless of how much effort we put into improving their readability.

We develop an approach to audit log views parameterised by the policies of data providers. Intuitively, given a policy $P := \{r_1, \ldots, r_n\}$ and a log view S, we define a transformation ν that takes P and returns the set of rules $V_P = \{v_1, \ldots, v_n\}$ such that each $v_i \in V$ denotes the violation of the corresponding rule r_i. To illustrate this, consider the rule r_2 in Fig. 3. By applying the transformation ν, the following violation is generated:

$v_2 :=$ `( allow, RFID-Reader, *, * )`.

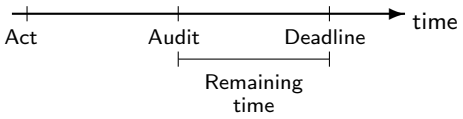


Fig. 6. Condition leading to an amber semaphore

This denotes that the collection of attributes through RFID readers is allowed, thereby contradicting the original desire of the data provider.

With V_P at hand, we then search for violations in the log view of the corresponding data provider. To this end, we define the pinpoint relation $\triangleright$ between views and the set of violations V_P such that $S \triangleright v_i$ if v_i can be pinpointed, i.e. detected, in S. If there is a $v_i \in V_P$ such that $S \triangleright v_i$, then there is an execution of the system that violates r_i and, in consequence, the policy P. In contrast, if there is no such v_i, such that $S \triangleright v_i$, then a violation of P can be ruled out. Technical details are found in [2].

We employ a semaphore notation to make the result of audit evident to the pertinent data provider. In this case, red obviously stands for a violation of some rule, while green denotes the compliance with a policy. An amber semaphore indicates that some obligation-based rule could not be pinpointed and therefore stands for a warning. Such a warning is triggered whenever a log view S is audited before the deadline of a pending obligation, as illustrated in Fig. 6.

A log view, together with the corresponding audit analysis, constitutes a privacy evidence. In the case of a violation, an individual may click over the semaphore and obtain details on which rules have been violated as well as the entries that led to this result. A similar procedure can be carried out when the semaphore shows amber.

6 Discussion and Perspectives

Taking stock, in this manuscript we purport the thesis that a holistic notion of control for IMS encompasses not only the regulation of communicated (respectively, collected) attributes, but also the supervision of adherence to stated policies. While this understanding of control as regulation and (at least the option of) supervision is prevalent in the common language, to our knowledge it has not been considered in the context of IMS. We firmly believe that the investigation of approaches to realise such forms of control is the next milestone towards the development of IMS to cope with the privacy challenges of dynamic systems.

portance of supervision as a distinguishing factor for future IMS and privacy-aware (dynamic) systems.

References

1. R. Accorsi. On the relationship of privacy and secure remote logging in dynamic systems. In S. Fischer-Hübner, K. Rannemberg, L. Yngström, and S. Lindskog, editors, *Proceedings of the 21st IFIP TC-11 International Security Conference: Security and Privacy in Dynamic Environments*, volume 201 of *International Federation for Information Processing*, pages 329–339. Springer-Verlag, 2006.
2. R. Accorsi and M. Bernauer. On privacy evidence for UbiComp environments – Broadening the notion of control to improve user acceptance. In A. Bajart, H. Muller, and T. Strang, editors, *Proceedings of the 5th Workshop on Privacy in UbiComp*, pages 433–438, 2007.
3. R. Accorsi and A. Hohl. Delegating secure logging in pervasive computing systems. In J. Clark, R. Paige, F. Pollack, and P. Brooke, editors, *Proceedings of the 3rd International Conference on Security in Pervasive Computing*, volume 3934 of *Lecture Notes in Computer Science*, pages 58–72. Springer Verlag, 2006.
4. M. Casassa-Mont, S. Pearson, and P. Bramhall. Towards accountable management of privacy and identity. In E. Snekkenes and D. Gollmann, editors, *Proceedings of the European Symposium on Research in Computer Security*, volume 2808 of *Lecture Notes in Computer Science*, pages 146–161. Springer-Verlag, 2003.
5. M. Froomkin. The death of privacy? *Stanford Law Review*, 52(5):1461–1543, May 2000.
6. A. Hohl. *Traceable Processing of Personal Data in Remote Services Using TCG*. PhD thesis, University of Freiburg, 2006.
7. S. Jajodia, M. Kudo, and V. Subrahmanian. Provisional authorizations. In A. Ghosh, editor, *E-Commerce Security and Privacy*, pages 133–159. Kluwer Academic Publishers, 2001.
8. E. Kenneally. Digital logs – Proof matters. *Digital Investigation*, 1(2):94–101, June 2004.
9. G. Müller. Privacy and security in highly dynamic systems. *Communications of the ACM*, 49(9):28–31, September 2006.
10. A. Pfitzmann. Multilateral security: Enabling technologies and their evaluation. In G. Müller, editor, *Proceedings of the International Conference on Emerging Trends in Information and Communication Security*, volume 3995 of *Lecture Notes in Computer Science*, pages 1–13. Springer-Verlag, 2006.
11. A. Pretschner, M. Hilty, and D. Basin. Distributed usage control. *Communications of the ACM*, 49(9):39–44, September 2006.
12. S. Sackmann, J. Strüker, and R. Accorsi. Personalization in privacy-aware highly dynamic systems. *Communications of the ACM*, 49(9):32–38, September 2006.
13. J. Strüker. Der gläserne Kunde im Supermarkt der Zukunft. *Wirtschaftsinformatik*, 49(1):59–62, January 2007.

JOURNAL OF INFORMATION SYSTEMS
Vol. 24, No. 1
Spring 2010
pp. 91–112

American Accounting Association
DOI: 10.2308/jis.2010.24.1.91

Continuous Auditing in ERP System Environments: The Current State and Future Directions

John R. Kuhn, Jr.
University of Louisville

Steve G. Sutton
University of Central Florida

ABSTRACT: Recent research has focused heavily on the practicality and feasibility of alternative architectures for supporting continuous auditing. In this paper, we explore the alternative architectures for continuous auditing that have been proposed in both the research and practice environments. We blend a focus on the practical realities of the current technological options and ERP structures with the emerging theory and research on continuous assurance models. The focus is on identifying the strengths and weaknesses of each architectural form as a basis for forming a research agenda that could allow researchers to contribute to the future evolution of both ERP system designs and auditor implementation strategies. There are substantial implications and insights that should be of interest to both researchers and practitioners interested in exploring continuous audit feasibility, capability, and organizational impact.

Keywords: continuous assurance; continuous auditing; continuous monitoring; enterprise systems; enterprise resource planning (ERP) systems; enterprise risk management; embedded audit modules; management control layer.

I. INTRODUCTION

Over the past 20 years, the discourse on the need for, and ability to deliver, continuous auditing of business information has slowly gained momentum. Today, the concepts of continuous auditing have reached the instantiation stage, becoming a key element in many internal audit departments' risk monitoring strategies. Additionally, continuous auditing is increasingly under consideration as a tool to augment the external audit. This progression has included the evolution of architecturally different methodologies for approaching continuous auditing in computerized environments, primarily embedded audit modules (EAM), which are software modules embedded in an information system (i.e., built into) to monitor activities in such systems (Groomer and Murthy 1989), and a monitoring control layer (MCL), which is an external software module that operates independently of the information system to be monitored but is linked into the system and/or its underlying database to provide a similar level of monitoring (Vasarhelyi et al. 2004). A key component of the discourse in recent years has been focused on the advantages,

Published Online: March 2010

limitations, and feasibility of the two dominant approaches (Alles et al. 2006; Groomer and Murthy 2003; Kogan et al. 1999; Kuhn and Sutton 2006; Vasarhelyi 2006; Vasarhelyi et al. 2004; Woodroof and Searcy 2001).

The constantly evolving state of corporate governance fueled by organizations' focus on strategic enterprise risk management has moved this debate to a focus on practicalities. Continuous auditing tools (EAM and MCL) are quickly becoming key components of overall corporate governance and compliance efforts. Many are designed for specific business processes, enterprise resource planning (ERP), legacy system control settings, transaction processing, and IT processes, etc., that taken together provide the foundation for a more holistic governance, risk management, and compliance (GRC) landscape supporting the entire enterprise (AMR Research, Inc. [AMR] 2008). Continuous auditing applications are being used to support GRC activities across business functions, departments, and IT platforms. As such, third-party software vendors (i.e., non-ERP developers) such as Approva have created additional applications to support enterprise-wide governance interlinking and sub-applications integration designed for monitoring of specific systems and processes. Features of the enterprise-wide governance applications include functionality that supports the identification and management of the varied GRC initiatives across a company (SOX 404 compliance, IT governance, corporate social responsibility, etc.); identification, assessment, categorization, and prioritization of risks; and tracking of key performance and risk indicators in an integrated fashion (e.g., dashboard reporting). The continuous auditing movement has evolved over time, expanded in scope and breadth, and continues to progress forward into unchartered territory.

The ensuing discussion presents a brief history of the origins of continuous auditing and continuous assurance; the development of continuous auditing applications and various alternatives available to companies and auditors; the current state of continuous monitoring and assurance of ERPs in practice; and where we see continuous auditing and overall assurance heading in the future in order to increase the clarity of the current state and provide suggestions for future research that would allow academic researchers to support and lead the growth and evolution. Our perspectives are founded in deep practical experience with ERP systems placed within the context of the evolving theory and research on continuous auditing.[1]

As the push for continuous auditing moves forward, it is important to see where we have been, where we are, and where we are headed. The current discussion contributes to the evolving literature on continuous auditing by examining the *technical* characteristics of continuous auditing through traditional EAM and MCL methods, a modified EAM approach, and newer enterprise-wide/cross-platform applications; the advantages and limitations of alternative methods; and contemporary developments. Of particular interest are the technological advances necessary to support initiatives such as XBRL data tagging and enterprise-wide GRC. Companies of varying sizes and views toward continuous auditing can and will need to choose continuous auditing functionality that supports their individual goals and desires—be it through EAM, MCL, a hybrid approach, or some new adaptation that has yet to evolve.

The remainder of the paper consists of four sections. The first section presents a brief history of the continuous auditing movement; characteristics of various architectural approaches; and advantages/limitations of the approaches from a technical and practical perspective—specifically, real-time monitoring and reporting, complexities of design and maintenance, alarm floods, client

[1] Our views are influenced in part by the first author's personal ERP audit experiences. Prior to entering academia in 2005, the author spent ten years in practice as a Big 4 IT auditor (managed the SAP audit of WorldCom during the fraud restatement) and Financial Systems Manager (managed an SAP implementation), and as an Internal Audit Manager at Siemens Corp.—auditing, evaluating, and implementing continuous auditing functionality both pre- and post-Sarbanes-Oxley 404.

independence, and external auditor legal liability. The second section provides an overview of existing continuous auditing software developed by third-party vendors and ERP developers—primarily, a look at SAP (ERP developer) and Approva (third-party) ventures into enterprise-wide continuous auditing to support overall GRC. The third section discusses continuous monitoring and auditing issues in need of research at both the application and enterprise-wide level that require a variety of research methods. The final section provides a brief summary of this review on the current status and future directions of continuous audit integration with ERP systems.

II. CONTINUOUS AUDITING BACKGROUND AND TYPES OF SYSTEMS

Beginnings of Continuous Auditing and Development of Embedded Audit Modules

ERP systems designed for the processing of business transactions traditionally have been built upon a core database management system (DBMS). Groomer and Murthy (1989) raised the awareness and interest of the accounting research community in the notion of EAM as a viable approach to supplement and enhance the audit of companies with increasingly complex computer-based accounting systems relying on DBMS. They described an approach to continuous auditing of database-driven accounting applications using EAMs to address the unique control and security aspects of database environments. They defined EAM as "modules (code) built into application programs that are designed to capture audit-related information on an ongoing basis." Braun and Davis (2003) similarly define EAM as follows:

> This technique involves the auditor inserting an audit module in the client's application that will identify transactions that meet some pre-specified criteria as they are being processed … Often, these modules are designed in such a way that they can be turned on and off, reducing costs but also reducing coverage. (Braun and Davis 2003, 726)

Vasarhelyi and Halper (1991) advanced the work of Groomer and Murthy (1989), developing the Continuous Process Audit Methodology (CPAM) and illustrating the design and functions of an EAM in a hypothetical customer billing system. The authors subsequently tested a prototype continuous auditing system they created based on CPAM for use with three large financial systems. The prototype results indicated a deeper and more reliable level of audit examination was achieved.

The early works of Groomer and Murthy (1989) and Vasarhelyi and Halper (1991), in conjunction with the joint report on continuous auditing sponsored by the Canadian Institute of Chartered Accountants and American Institute of Certified Public Accountants (CICA/AICPA 1999), spawned a stream of continuous auditing research. From an architectural standpoint, a good portion of the efforts adhered to the methodology from these two early works by focusing on EAM as the underlying technological approach (Groomer and Murthy 2003; Debreceny et al. 2003; Murthy 2004; Debreceny et al. 2005; Chen et al. 2007; Loh and Jamieson 2008).

Debreceny et al. (2005) defines a set of essential characteristics for successful continuous monitoring applications: (1) an end-user environment that allows the auditor to establish a set of queries to test transaction integrity constraints either from a pre-defined suite of queries, the modification of the attributes of pre-defined queries, or by the creation of new queries by the construction of simple scripts; (2) a process for registration and scheduling of these queries; (3) a method for running these queries against the flow of transactions for violations either continuously or temporally; (4) a capacity for reporting violations electronically; and (5) an ability to copy the transaction details of violations to secondary storage. We elaborate on these as we create a comparison of characteristics for a variety of continuous auditing application design approaches. But first, clarification of some related terms will help. *Continuous monitoring*, also often referred to as *continuous auditing*, consists of the analysis of data on a real- or near real-time basis against a set of predetermined rule sets. Henrickson (2009) suggests the delineating determinant between the

two can be viewed as monitoring being a management responsibility and auditing being an auditor's responsibility. *Continuous reporting*, on the other hand, is the continuous reporting of information about defined phenomena and in a continuous auditing context includes the notification of rule violations occurring as a result of continuous monitoring. Finally, at the macro level sits *continuous assurance*, as noted by Alles et al. (2002, 128):

> [Continuous auditing] is best described as the application of modern information technologies to the standard audit products … Continuous auditing is another step in the path of the evolution of the financial audit from manual to systems-based methods … By contrast, continuous assurance sees continuous auditing as only a subset of a much wider range of new, nonstatutory products and services that will be made possible by these technologies.

All of the approaches (monitoring, auditing, assuring) require the same basic technical capabilities. There are alternative approaches to delivering these capabilities, however, other than EAM. Before considering these alternatives, we review the design and implementation characteristics of EAM.

First, with EAM the programming code for the audit procedures/tasks is developed and implemented *inside* the walls of the target application (i.e., embedded) using the programming language of the application itself. For instance, SAP developed a unique programming language called ABAP for specific use with their core ERP software. If, for instance, a company wishes to build an EAM audit check that verifies every invoice has a corresponding purchase order and goods receipt with identical quantity and dollar amounts within a certain threshold range (i.e., commonly referred to as the three-way matching process), programmers for the company can create an ABAP program and literally "plant it" inside SAP alongside the programs delivered with SAP. This new program is considered "non-native code" in the respect that it is added to SAP afterward. Second, EAM audit functionality can evaluate transactions against programmed audit criteria *live* (i.e., real-time monitoring) as they happen in the application. That does not necessarily mean the EAM module *has* to run constantly, but it *can*, if so desired. Third, EAMs include reporting functionality that notifies predetermined individuals through any of a number of options, including emails, pager notifications, system reports, etc., of any transactions violating the audit procedures. Due to the real-time monitoring functionality, EAM consequently offers real-time reporting. In our three-way matching example, if an invoice was processed and flagged for payment but the invoice amount exceeded the original purchase order by a predetermined threshold (say 1 percent), then an instantaneous notification (referred to as an *alarm*) would be sent to individuals in the internal audit department or external audit firm (or both) depending on what party relies on the EAM. Fourth, storage of the alarms for data retention purposes resides in the same databases/database structures with normal transactions so regularly scheduled application backups capture the EAM auditing data. This also allows the auditor to handle real-time alerts in a batch mode, if so desired. We explore the rationale for such a decision later.

Alternative Technical Architectures to Traditional EAM

The traditional EAM model prescribes coding of audit procedures directly into the auditee's host system to facilitate real-time monitoring and reporting of transactions and system settings. However, certain characteristics of a traditional EAM instantiation may not be the best fit for every company and situation. Therefore, the question arises as to the alternatives available to companies and auditors for continuous auditing implementation. Can the EAM model developed in extant research be modified? Are other architectures more feasible under certain circumstances? Or, are the various continuous auditing architectural strategies situational and, therefore, amenable to co-existence within organizations?

EAM Ghosting

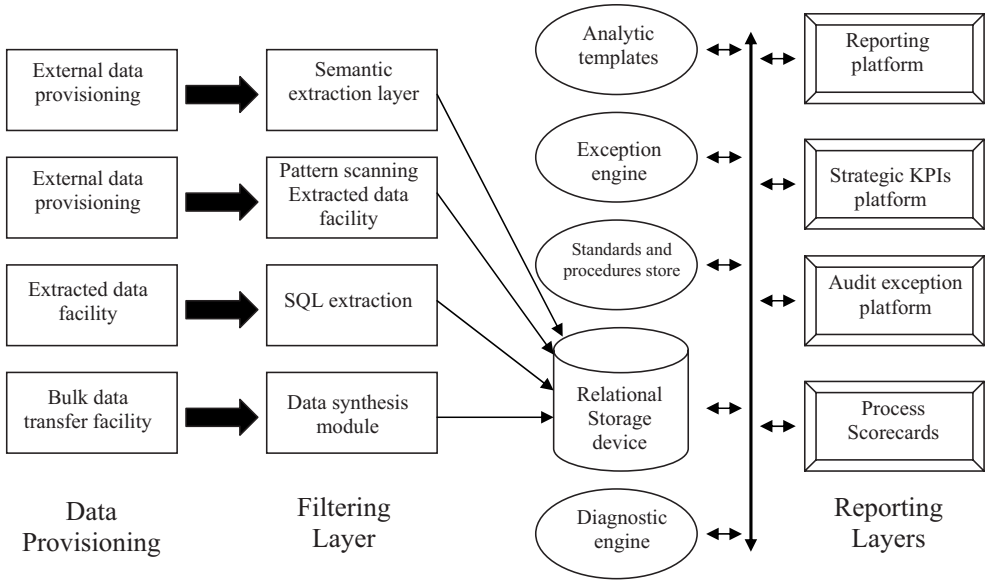
System "ghosting" offers an opportunity to benefit from the advantages of EAM yet implement, operate, and maintain audit functionality outside the production system of a company keeping the audit program separate from the live system. Ghosting entails operating a "copy" of an entire system on separate hardware, including data and system settings, in a real-time fashion similar to an instantaneous fail-over common to disaster recovery and business continuity planning procedures and redundant firewall systems. The EAM functionality would reside in the ghosted copy where there would be no risk of affecting live transaction processing occurring within the production environment.

Two slight variations of EAM Ghosting also exist. Many companies operating large-scale ERP systems utilize a stream of copies of the production environment used for development and change management. Separate systems exist for development (DEV), quality assurance (QAS), and production (PRD), generally hosted on different physical servers that communicate through transport management software (VMware 2009). Changes intended for production will first be configured and tested in DEV by IT personnel, then migrated to QAS for end-user testing before transport to PRD. Typically, the DEV instance contains production data a few days to months old, but with current systems settings. QAS data will be closer in time to PRD with the same system settings. Both variations take advantage of the fact that the DEV and QAS physical servers experience significantly less usage than PRD. For the first variation, companies can segregate a physical server such as the one housing QAS into two partitions, one supporting QAS and the other supporting a mirror-image of PRD updated nightly (rather than real-time) that contains the EAM code. The primary advantage of this approach over standard EAM Ghosting is that IT personnel do not have to worry that real-time processing is updated instantaneously in the system housing the EAM. Second, virtualization technology provided by companies like VMware can take advantage of unused physical hardware space and create an independent virtual machine on the unused QAS server that ghosts the production system but contains the EAM programs. Virtualization requires less physical hardware space and less memory to operate than creating a separate partition on an existing server.

Monitoring Control Layer

Vasarhelyi et al. (2004) introduce an alternative continuous auditing system architecture to EAM referred to as the Monitoring and Control Layer (MCL). MCL can be viewed as the next stage in the evolution of continuous auditing application design—not necessarily one subsuming EAM but more as an alternative to cater to different circumstances. The MCL approach takes the continuous auditing system and "hooks" it into the client's existing enterprise system, generally using a middleware layer to integrate loosely coupled applications such as ERP systems, legacy systems, and web-based applications (e.g., supply chain management, customer relationship management, etc.). The main elements of the MCL architecture consist of: (1) data capture layer; (2) data filtering layer; (3) relational storage; (4) measurement standards layer; (5) inference engine; (6) analytic layer; (7) alarms and alerting layer; and (8) reporting platform as depicted in Figure 1 (Vasarhelyi et al. 2004). Essentially, the continuous auditing system (designed with a simple user interface and underlying database reusable for multiple clients) resides outside the client's network environment (for external auditors) and is controlled by the auditor. The continuous auditing system receives periodic interfaces of client data as determined necessary by the auditor (i.e., not real-time) that are processed against a predefined rule-set of audit procedures inside the continuous auditing application, which is physically and virtually outside the client's audited system (unlike EAM). Any violations as defined by the rule-set trigger an automatic alert to the auditor. These alerts are stored inside the continuous auditing application under direct control by the auditor, not inside the databases supporting the client's ERP system.

FIGURE 1
The Architecture of the MCL Layer
Continuous Monitoring and Control Platform



Source: Vasarhelyi et al. (2004).

Recent research efforts by Kuhn and Sutton (2006), Alles et al. (2006), and Alles et al. (2008) offer evidence of the viability of an MCL continuous auditing approach in an ERP environment. As an example of how fraudulent activities at WorldCom could have been detected earlier, Kuhn and Sutton (2006) lay out the design specifications for integrating a specific set of metrics appropriate for the WorldCom financial reporting system that could have been used to continuously monitor transactions. The study demonstrates how financial transaction data (i.e., journal entries) can be extracted from the client database layer without any direct processing inside the client system. Both Alles et al. (2006) and Alles et al. (2008) document a functioning system prototype developed at Siemens for the continuous monitoring of business process controls and the detection of exceptions to those controls. The prototype demonstrates the use of ABAP (SAP's unique language programming) to extract the business process control data from Siemens' ERP system. The two studies differ on several key aspects. Kuhn and Sutton (2006) *design* continuous auditing procedures for the testing of *financial transactions* based on an *historical case* of financial reporting fraud. Alles et al. (2006) and Alles et al. (2008), on the other hand, *implement* a continuous auditing system for the testing of *internal controls* in a *live* environment.

Table 1 presents general characteristics for continuous auditing systems and how various architectural types compare on those traits. The remainder of this section focuses on analysis of the issues and concerns associated with each approach, including some that all approaches share.

TABLE 1
Characteristics of Continuous Auditing System Types

Characteristic	EAM	EAM Ghosting	MCL
Contains predefined audit rule-set.	Yes	Yes	Yes
Location of audit rule-set and program.	Inside target system	Outside target system, inside client network	Outside target system; outside client network if external auditor; inside client network if internal auditor
Requires installation of additional hardware to implement.	No	Possibly	Yes
Performs real-time monitoring.	Yes	Yes	No
Performs real-time reporting.	Yes	Yes	No
Automatically notifies auditor of rule violations.	Yes	Yes	Yes
Location of stored rule violations.	Target system database	Database external to target system, inside client network	Database external to target system; outside client network if external auditor; inside client network if internal auditor
Owner of system resources.	Client	Client	Client if internal auditor, Audit firm if external auditor

Limitations of Continuous Auditing Applications

Debreceny et al. (2005) provide an overview of several limitations in the use of EAM along with observations regarding the lack of perceived demand for EAM capability. In the Alles et al. (2002) discussion on the feasibility and economics of continuous assurance, the authors raise some concerns with EAM for external auditors. They question whether EAMs designed by external auditors and incorporated into ERP systems ultimately transform the ERP into an "unauditable" system due to a perceived or actual lack of independence. A technical dependence invariably exists between the assuring system (EAM) and assured system (ERP), and the auditor may well be perceived partially responsible for the client's enterprise system. As Alles et al. note, questions such as these can only be answered through theoretical and empirical research. The following discussion expands upon the limitations of EAM, EAM Ghosting, and MCL in greater depth in an effort to improve the understanding of the evolution of continuous auditing and related applications, specifically why ERP vendors were slow to develop EAM functionality.

Technical Concerns

ERP systems, by the nature of their integrated processes, require significant information system resources to operate at an optimal level. As Debreceny et al. (2005) noted, Vasarhelyi and Halper (1991) expressed concerns related to the computing resources EAMs require to monitor transactions and system settings in real-time. Those additional resource requirements risk slowing

lem, that of independence. Both scenarios would result in knowledge of the audit procedures and ultimately could be subject to client manipulation. The client could potentially utilize the inside information on the auditor's monitoring procedures to conduct fraudulent activities in unaudited areas or design suspect transactions configured intentionally to not trigger alerts. In short, the existing independence requirements and prescribed procedures for unannounced audit visits are in place to assure the auditor has the opportunity to monitor and search for fraudulent activity without the client having *a priori* knowledge of what will be tested and observed. Sacrifice of this ability seems atheoretical to the underpinnings of audit independence.

Legal liability. A major deterrent to auditors assuming many of the responsibilities often seen desirable by the public (e.g., fraud) has been the risk of legal liability. While the client may be apprehensive about an auditor having access and making changes within their system or simply guiding the implementation of EAM modules within the system, the auditor would likewise be expected to have some trepidation over the risk involved in making such changes to a client's system. Systems are core to an entity's ability to continue its operations and compete in the marketplace. The threat to systems from interacting with trading partners has become severe enough that some powerful corporations such as Wal-Mart have required trading partners that interact with their systems through e-commerce to sign a contractual agreement that the trading partner is responsible for all costs related to damages and repairs to Wal-Mart's systems along with all costs incurred from lost business should their system be infected or otherwise damaged by the trading partner linkage (Gerhardt 2002). It does not seem unreasonable to assume a client organization would want a similar level of risk reduction from the auditor if the auditor is going to be put into a position where damage to the client's system may occur.

Regardless of whether the client and auditor have signed a damage responsibility agreement, should any of the modifications implemented by the auditor adversely affect the processing of client business transactions, the client would likely take legal action against the auditor in order to recover damages incurred. This threat of litigation alone would likely be sufficient to cause many public accounting firms to refuse to undertake continuous auditing of client systems using an EAM strategy.

The GRC Software Market

The massive amounts of resources (man-hours and money) expended to comply with SOX during the first years have driven companies to seek opportunities to streamline the ongoing compliance process. Continuous auditing applications offer the ability to strengthen the internal control environment and provide efficiencies in the overall compliance activities. However, as Debreceny et al. (2005, 23) noted, ERP software vendors generally "believe customers are unwilling to pay a premium for a function that is not perceived as mission-critical" and, therefore, only limited (if any) built-in functionality exists. The disconnect between ERP vendors and their customers may lie in the fact that accelerated filers initially grossly underestimated the resources required to comply in 2004 and the ERP vendors did not receive any type of substantial call for the additional functionality early in the SOX compliance process.

Companies are now looking for ways to reduce the level of manual effort and the related consulting costs incurred the early years of SOX 404 compliance through technology applications. Furthermore, recent PCAOB Auditing Standard No. 5 guidance that recommends external auditors rely more on internal audit work provides an even greater impetus for companies to find alternative solutions for performing automated testing. Independent, third-party software developers have seized the opportunity and developed SOX compliance tools that monitor the effectiveness of internal controls in the financial systems along with the system settings that facilitate control processes; the vendors target these applications directly at internal audit departments and other

helping to understand how well organizations assimilate and leverage continuous auditing systems. On the other hand, the unique nature of continuous auditing systems as a control mechanism may also yield advancements in existing theories of assimilation that expand the understanding of the applicability to a broader range of systems.

- What are the organizational factors that lead an organization to adopt continuous audit technologies?
- What are the organizational factors that drive strong assimilation of continuous audit technologies?

The research challenges put forth in this paper likely only scratch the surface. Our understanding of the impact of continuous auditing systems at this point in time is rudimentary and the opportunities for research that can have a significant impact are great.

IV. CONCLUSION

The development and pervasive use of ERP systems provides the critical infrastructure necessary for the effective evolution of the assurance function from a periodic event to an ongoing process through the integration of continuous auditing applications. Two principal system architectures for the development and continuous auditing applications have emerged in research literature. The Embedded Audit Module methodology developed early on in the domain's existence integrates continuous auditing functionality internally within the system of concern and operates in a truly real-time, continuous mode. Subsequently developed as an alternative to EAM, the Monitoring and Control Layer methodology focuses on an external module that interfaces with the system of concern to retrieve scheduled interfaces of selected data. Each of the continuous auditing approaches offers distinct advantages over the other.

Debreceny et al. (2005) examine the nature of EAM functionality and the relationship with ERP systems, perceived reasons for the lack of widespread adoption of EAM, and potential directions to increase the viability of the EAM approach to continuous auditing. In order to add additional clarity to the discussion, our study expands upon the limitations of EAM noted by Debreceny et al. (2005) and also explores the limitations of alternative continuous auditing approaches such as EAM ghosting and MCL. These limitations are examined from both technical and practical perspectives, as well as also examining the issues specifically from an audit perspective which carries major concerns of design and maintenance, client independence and auditor legal liability. We examined alternative strategies for implementation of continuous auditing, considering both (1) a modified EAM approach through the use of "ghosting" technology and (2) the MCL methodology. At present, MCL methodology appears to offer the most used alternative, as evidenced by both the recent increased attention in academic research (Alles et al. 2006; Kuhn and Sutton 2006; Alles et al. 2008) and the exclusive use of MCL for continuous auditing products developed by third-party vendors (AMR 2008).

As Debreceny et al. (2005) note, no prior research existed on EAM support within ERP systems at the time of their study. Only two additional research projects have surfaced since that point that examine continuous auditing in an ERP environment (i.e., Kuhn and Sutton 2006; Alles et al. 2006; Alles et al. 2008). Given the pervasiveness of ERP environments, additional research is necessary to advance the awareness, relevance, and practicality of ERP continuous auditing. In the course of discussing the issues fundamental to the use of EAM versus MCL, we have outlined a number of key research issues. The opportunities for research are plentiful and the need even greater when considered in light of wide ranging issues related to the potential demand for continuous monitoring and continuous auditing (e.g., Daigle and Lampe 2004, 2005), to the consideration of external auditor willingness and capability to implement and utilize continuous

auditing strategies, to the transference and increasing reliance of internal control work performed by internal auditors, to the need for additional instantiations of continuous auditing applications in "real" environments (e.g., Alles et al. 2006; Kuhn and Sutton 2006), and to the examination of innovative integrated IT platforms that cross technologies. Exploration has only just begun on understanding the behavioral effects on an organization's employees and managers (e.g., Hunton et al. 2008), but these issues are also critical to anticipating the impact as these systems become more widespread. Opportunities abound for research facilitating the efficient and effective implementation and utilization of continuous auditing capabilities.

REFERENCES

Alles, M., A. Kogan, and M. A. Vasarhelyi. 2002. Feasibility and economics of continuous assurance. *Auditing: A Journal of Practice & Theory* 21 (1): 125–138.

———, G. Brennan, A. Kogan, and M. A. Vasarhelyi. 2006. Continuous monitoring of business process controls: A pilot implementation of a continuous auditing system at Siemens. *International Journal of Accounting Information Systems* 7 (2): 137–161.

———, A. Kogan, and M. A. Vasarhelyi. 2008. Putting continuous auditing theory into practice: Lessons from two pilot implementations. *Journal of Information Systems* 22 (2): 195–214.

AMR Research, Inc. (AMR). 2008. Enterprise performance management: 2008 landscape series. Available at: http://www.approva.net/assets/resources/AMR_Research_REPORT_21828-The_Governance,_Risk_Management,_and_Com.pdf.

Behn, B. K., D. L. Searcy, and J. B. Woodroof. 2006. A within firm analysis of current and expected future audit lag determinants. *Journal of Information Systems* 20 (1): 65–86.

Braun, R. L., and H. E. Davis. 2003. Computer-assisted audit tools and techniques: analysis and perspectives. *Managerial Auditing Journal* 18 (9): 725–731.

Canadian Institute of Chartered Accountants and American Institute of Certified Public Accountants (CICA/AICPA). 1999. *Continuous Auditing. Research Report*. Toronto, Canada: CICA.

Chatterjee, D., R. Grewal, and V. Sambamurthy. 2002. Shaping up for e-commerce: Institutional enablers of the organizational assimilation of web technologies. *Management Information Systems Quarterly* 26 (2): 65–89.

Chen, H. H., S. H. Li, S. M. Huang, and Y. C. Hung. 2007. The development and performance evaluation of a continuous auditing assistance system. *International Journal of Electronic Finance* 1 (4): 460–472.

Daigle, R. J., and J. C. Lampe. 2004. The impact of the risk of consequence on the relative demand for continuous online assurance. *International Journal of Accounting Information Systems* 5 (3): 313–340.

———, and ———. 2005. The level of assurance precision and associated cost demanded when providing continuous online assurance in an environment open to assurance competition. *International Journal of Accounting Information Systems* 6 (2): 129–156.

Debreceny, R. S., G. L. Gray, W. L. Tham, K. Y. Goh, and P. L. Tang. 2003. The development of embedded audit modules to support continuous monitoring in the electronic commerce environment. *International Journal of Auditing* 7 (2): 169–185.

Debreceny, R. S., ———, J. Jun-Jin Ng, K. Siow-Ping Lee, and W. Yau. 2005. Embedded audit modules in enterprise resource planning systems: Implementation and functionality. *Journal of Information Systems* 19 (2): 7–27.

Deloitte. 2007. Security user segregation of duties—SAP GRC. Available at: http://www.Deloitte.com/dtt/case_study.

Dowling, C. 2009. Appropriate audit support system use: The influence of auditor, audit team and audit firm factors. *The Accounting Review*.

Ernst & Young. 2006. SEC and PCAOB roundtable discussion on implementation of internal control reporting provisions: Year Two. *Emerging Trends in Internal Controls* (May): 1–12.

Fogarty, J. 2005. *Continuous Auditing: Can it Become a Reality and What is its Impact on Auditing Standards*. Rutgers Tenth Continuous Auditing and Reporting Symposia. Newark, November.

Gerhardt, K. W. 2002. *The Emerging Role of the Information Systems Function in Modern Organizations:*

Journal of Information Technology (2016) 31, 1–31

palgrave-journals.com/jit/

Research article

Determinant factors of cloud-sourcing decisions: reflecting on the IT outsourcing literature in the era of cloud computing

Stephan Schneider, Ali Sunyaev

University of Cologne, Cologne, Germany

Correspondence:
Ali Sunyaev, University of Cologne, Pohligstraße 1, 50969 Cologne, Germany.
Tel: +49-221-470-5397;
Fax: +49-221-470-5386;
E-mail: sunyaev@wiso.uni-koeln.de

Abstract
Cloud computing (CC) is an emerging form of IT outsourcing (ITO) that requires organizations to adjust their sourcing processes. Although ITO researchers have established an extensive knowledge base on the determinant factors that drive sourcing decisions from various theoretical perspectives, the majority of research on cloud-sourcing decisions focuses on technological aspects. We reviewed the CC and ITO literature and systematically coded the determinant factors that influence sourcing decisions. We show that most determinant factors of sourcing decisions in the ITO context remain valid for the CC context. However, the findings for some factors (i.e., asset specificity, client firm IT capabilities, client firm size, institutional influences, and uncertainty) are inconclusive for the CC and ITO contexts. We discuss how the peculiarities of CC can explain these inconclusive findings. Our results indicate that CC researchers should draw from research on ITO decision making but re-examine ITO concepts in the light of the peculiarities of CC, such as the differences between software and infrastructure services, the self-service procurement of cloud services, or the evolving role of IT departments. By summarizing determinant factors of cloud-sourcing decisions for consideration in future research, we contribute to the development of endogenous theories in the IS domain.
Journal of Information Technology (2016) **31,** 1–31. doi:10.1057/jit.2014.25;
published online 11 November 2014
Keywords: cloud computing; IT outsourcing; sourcing decision; determinant factors; adoption; literature review

Introduction

Cloud computing (CC) changes how organizations manage their IT landscape, challenges traditional IT governance approaches, and requires organizations to adjust their sourcing processes (Yanosky, 2008; Armbrust *et al.*, 2010; Winkler and Brown, 2013; Ragowsky *et al.*, 2014). With CC, organizations can gain on-demand network access to a shared pool of managed and scalable IT resources, such as servers, storage, and applications (Mell and Grance, 2011). Because IT sourcing decisions entail substantial economic and strategic risks (Martens and Teuteberg, 2009; Benlian and Hess, 2011), managers must have extensive judgment and insight regarding organizational structures, interdependencies, processes, and habits to thoroughly comprehend decision alternatives and the set of required structural choices (McIvor and Humphreys, 2000; Cullen *et al.*, 2005; Moses and Åhlström, 2008; Aubert *et al.*, 2012). However, empirical insight into cloud-sourcing decisions remains scarce (Yang and Tate, 2012).

CC is an evolution and specific form of IT outsourcing (ITO); thus, the extensive body of research on ITO provides a valuable basis for investigating cloud-sourcing decisions. During the last two decades, IS researchers have applied various economic, strategic, organizational, and social theories to identify determinant factors that drive ITO and have produced a considerable body of knowledge on ITO decision making (Dibbern *et al.*, 2004; Lacity *et al.*, 2010). However, rather than leveraging this insightful body of knowledge on ITO, the majority of research on the determinant factors of cloud-sourcing decisions focuses on technological aspects of CC and their implications for decision making (e.g., Koehler *et al.*, 2010; Wu *et al.*, 2011; Brender and Markov, 2013; Gupta *et al.*, 2013; McGeogh and Donnellan, 2013).

client interface (e.g., a web browser). In the context of SaaS, users do not own, manage, or operate the underlying infrastructure, platform, or even individual application capabilities (Mell and Grance, 2011). Examples range from complex enterprise applications, such as SAP Business ByDesign, to office, email, and collaboration services, such as Google Apps.

Public clouds are owned, managed, and operated by external providers and are made available to the general public over the Internet (Armbrust *et al.*, 2010: 50), whereas *private* clouds are 'provisioned for exclusive use by a single organization' (Mell and Grance, 2011: 3) and are not available to the general public. However, given clouds' public availability, security and privacy have become important concerns in public CC (European Network and Information Security Agency, 2011; Jansen and Grance, 2011). Private clouds are 'easier to align with security, compliance, and regulatory requirements [than public clouds]' (Ramgovind *et al.*, 2010: 2), but they offer fewer benefits, for example, in terms of cost reduction and scalability. Mixed forms of private and public clouds aim to combine the benefits of both public and private clouds. *Community* clouds are 'controlled and used by a group of organizations that have shared interests, such as specific security requirements' (Marston *et al.*, 2011: 180), where 'its strengths and weaknesses fall between those of a private cloud and those of a public one' (European Network and Information Security Agency, 2011: 55). *Hybrid* clouds are a combination of public and private cloud services; 'typically, noncritical information is outsourced to the public cloud, while business-critical services and data are kept within the control of the organization' (Marston *et al.*, 2011: 180).

During recent years, CC has received enormous attention from academics and practitioners, resulting in a disparate collection of publications from research and practice. To synthesize the body of knowledge on CC, researchers have conducted literature reviews that provide a general overview of CC research (Yang and Tate, 2012), that focus on a specific domain (Ermakova *et al.*, 2013), or that focus on particular aspects, such as security (Khorshed *et al.*, 2012). However, this study is the first to survey and synthesize the literature on IT sourcing decisions, to discuss differences between findings on CC and ITO, and to derive determinant factors of cloud-sourcing decisions. To distinguish this literature review from previous literature reviews, an overview of existing literature reviews on CC is presented in Appendix A.

Cloud-sourcing decisions

The unit of analysis in this article is *determinant factors of cloud-sourcing decisions*. We define a *cloud-sourcing decision* as an organization's decision to adopt and integrate cloud services from external providers into their IT landscape, that is, the customer organization's assessment of CC offerings from one or more providers in any form of service model (IaaS, PaaS, SaaS) or deployment model (public, private, community, hybrid). We thereby explicitly exclude organizational (internal) activities related to data center virtualization and IT services provided by an internal IT department.

Concerning IT sourcing decisions, researchers have investigated a range of dependent constructs, including the 'intention to increase the level of SaaS adoption' (Benlian and Hess, 2011), 'ASP adoption intention' (Yao *et al.*, 2010), 'netsourcing decision' (Loebbecke and Huyskens, 2006), or the 'degree of IS outsourcing' (Ang and Straub, 1998). These examples highlight researchers' use of different constructs to denote adoption decisions of various sourcing options (e.g., CC, ASP, ITO). In our work, we follow Lacity *et al.* (2010, 2011) and use one meta-construct to subsume these constructs, that is, the *IT sourcing decision*.

Cloud computing as new form of ITO

ITO can be defined as 'the significant contribution by external vendors in the physical and/or human resources associated with the entire or specific components of the IT infrastructure in the user organization' (Loh and Venkatraman, 1992a: 9). Cost advantages, flexibility, and competitive advantages are possible benefits that have made ITO one of the most important strategic concepts in recent decades (Leimeister *et al.*, 2010) and an intensively studied field within IS research (Dibbern *et al.*, 2004; Lacity *et al.*, 2010).

Application service provision (ASP) is a 'form of outsourcing, specifically selective outsourcing, where firms rent packaged software and associated services from a third party' (Bennett and Timbrell, 2000: 196). Jayatilaka *et al.* (2003) extend this definition and define ASP as an IS application service that is offered, hosted, and managed by a vendor on a rental basis. Researchers argue that SaaS emerged as an advanced form of ASP (Heart, 2010; Benlian and Hess, 2011) and that CC 'has the same key attributes as the "standard" ASP model, and exposes the user to the same risks' (Schwarz *et al.*, 2009: 774). The definitions of ASP and SaaS as well as their similarities in strategic, technical, and economic opportunities and risks (Kern *et al.*, 2002; Schwarz *et al.*, 2009; Armbrust *et al.*, 2010; Marston *et al.*, 2011) support this proposition. Furthermore, both ASP and SaaS share similar business and pricing models (Weinhardt *et al.*, 2009). Given the purpose of this research and the similarities between ASP and SaaS, we consider SaaS to be an advanced form of ASP and do not further differentiate these two sourcing options.

CC and ITO share common characteristics and provide similar benefits to users; research findings on ITO adoption are thus applicable to CC adoption to a certain extent (Benlian and Hess, 2011; Malladi and Krishnan, 2012; Chen and Wu, 2013). However, certain peculiarities regarding CC distinguish it from ITO and therefore induce the need to re-examine the determinant factors of sourcing decisions for CC. For instance, compared with traditional ITO, CC induces a shift in task responsibilities during decision processes and self-service procurement, provides standardized services with a narrower scope, enables new scenarios of outsourcing and governance arrangements, and uses short-term usage-based contracts (Susarla *et al.*, 2010; Benlian and Hess, 2011; Malladi and Krishnan, 2012; Chen and Wu, 2013). Table 1 summarizes the major differences between CC and ITO.

To determine what we can learn from the rich body of research on ITO and to identify determinant factors of cloud-sourcing decisions, we use the arguments listed in Table 1 as analytical devices to discuss the differences identified in the literature between CC-related findings and ITO-related findings. Therefore, we first survey the literature and aggregate the existing research on IT sourcing decisions. The next section describes our literature search, selection, and coding methods.

Research method

In this section, we first describe the scope of the literature review and the criteria that we applied to identify the 88

Table 1 Comparison of cloud computing and IT outsourcing

	Cloud computing	*IT outsourcing*
Decision process	• SaaS: business department as key client • IaaS/PaaS: IT department as key client • Predominantly self-service • Vendor selection bound to product selection, product-based decision • Online trial evaluations • Task responsibilities shifted from provider to customer, for example, for request for proposal evaluation *vs* self-service evaluation	• Large outsourcing contracts with high strategic relevance, top management as key clients • Request for information/request for proposal • Vendor selection prior to decision on degree of outsourcing
Scope	• Standardized software (SaaS) or cloud infrastructures (IaaS/PaaS) created by the provider for an anonymous market • Role of the IT department as service integrator • Limited customization	• Custom-tailored IT services • Can include hardware, software, people, and processes (e.g., software development, datacenter operations, desktop maintenance, help desk operations)
Governance mode	• Enables new scenarios of outsourcing and governance arrangements due to the variety of service models (IaaS, PaaS, SaaS) and deployment models (private, public, community, hybrid) and combinations thereof • Enables the management of building blocks of IT, provided by external providers in the same way as they would be managed in-house • Ownership, mode, and degree partially predefined by the selected service and deployment model	• Individual configurations of ownership, mode, and degree
Ownership	• Outsourced assets totally owned by the provider and its providers	• Varies with type and degree of outsourcing • Totally owned by the customer • Partially owned by the customer • Totally owned by the provider
Mode	• Single vendor/client or multiple vendors/ clients	• Single vendor/client or multiple vendors/clients
Degree	• Selective outsourcing	• Total outsourcing • Selective outsourcing
Contractual mode	• Short term • Usage based • High degree of automation and scaling • Minimal up-front costs • Little possibility for negotiation, standardized terms of use	• Long term • Period based or project based • Individually negotiated • Pricing based on business metrics • Strategic partnerships for continuous and joint innovation
Environment	• Decentralized market • Volatile and immature market • Uncertain legal issues	• Outsourcing market is well established with numerous experienced providers
Broad network access	• Critical network dependence • Potential bottlenecks, slowdowns, and outages that neither the client nor the vendor can control	• Depends on the type of outsourcing (e.g., less critical for software development than for datacenter operations)
Resource pooling	• Multi-tenant virtualized applications • Common code stack • Provider-determined upgrade schedule	• None

Note: This table was constructed based on the following: (Loh and Venkatraman, 1992a; Lacity and Hirschheim, 1993a; Bennett and Timbrell, 2000; Kakabadse and Kakabadse, 2002; Kern *et al.*, 2002; Jayatilaka *et al.*, 2003; Morabito, 2003; Serva *et al.*, 2003; Dibbern *et al.*, 2004; Narayandas, 2005; Dhar and Balakrishnan, 2006; Pollock and Williams, 2007; Xin and Levina, 2008; Benlian *et al.*, 2009; Schwarz *et al.*, 2009; Susarla *et al.*, 2010; Benlian and Hess, 2011; Martens and Teuteberg, 2011; Marston *et al.*, 2011; Mell and Grance, 2011; Vetter *et al.*, 2011; Giessmann and Stanoevska, 2012; Malladi and Krishnan, 2012; Lacity and Willcocks, 2013; Sunyaev and Schneider, 2013; Wollersheim *et al.*, 2013; Ragowsky *et al.*, 2014).

relevant articles that were published before April 2014. We then explain the applied method to code and aggregate the determinant factors of IT sourcing decisions.

Scope of the review

To ensure a high-quality literature review, we followed the guidelines by Webster and Watson (2002). This structured approach assumes that the major contributions in a research field are primarily found in journals of high reputation and quality. Therefore, we considered the Senior Scholars' Basket of Journals (Association for Information Systems, 2011) and the top 50 journals (including selected ACM/IEEE Transactions) of the AIS journal ranking (Association for Information Systems, 2005). To include the latest research on CC but still focus on high-quality contributions, we broadened the scope by additionally considering leading conferences of the IS community. The peer selection of outlets considered in this literature review consisted of 68 journals and 3 conferences (see Appendix B). We aimed to gather all contributions dedicated to CC or related paradigms that focus on sourcing decisions. We therefore searched publications by title, keywords, and abstract using the following list of keywords: (Cloud OR IaaS OR PaaS OR SaaS OR XaaS OR 'Infrastructure as a Service' OR 'Platform as a Service' OR 'Software as a Service' OR 'IT service' OR 'Application Service' OR ASP OR Outsourcing) AND (Adoption OR Assimilation OR Choice OR Decision OR Determinant OR Diffusion OR Driver OR Infusion OR Inhibitor OR Option OR Select OR Usage OR Use).

Inclusion and exclusion criteria

We included only completed peer-reviewed research articles that empirically investigate organizational sourcing decisions in the context of CC, ASP, or ITO and excluded conceptual articles, news articles, and reviews of prior research. To transfer findings from the ITO context to the CC context and to discuss what the literature on cloud-sourcing decisions can learn from the literature on ITO decisions (i.e., which findings are transferable to the CC context and which findings might require reconsideration), we had to limit the context of the ITO articles included to a common denominator. Therefore, studies concerning ITO were included only if they investigate sourcing decisions that are similar to sourcing decisions for cloud services. We used the following inclusion criteria for the selection process:

Applicability to CC

When considering articles concerning ITO, we referred to the descriptive ITO framework of de Looff (1995), who describes which parts of an organization's IT function are outsourced according to three dimensions:

- functional information systems (the business process that a system supports or controls, such as customer relationship management or order scheduling);
- components (hardware, software, data, personnel, and procedures); and
- activities (planning, development, implementation, maintenance, and operation).

For example, an organization can outsource the *development* of the *software* for a *customer relationship management system*. To include only outsourcing types that are comparable to sourcing decisions concerning one of the three cloud service models (IaaS, PaaS, and SaaS), we included only articles that investigate *selective outsourcing* of at least the *maintenance and operation* of the *hardware or software* of *any functional information system* to an *external provider* while keeping the business process itself in-house. Therefore, we excluded, for example, research focusing on software development, help desk, desktop maintenance, or business process outsourcing, as these types of outsourcing are specific in terms of the associated managerial problems, required capabilities, and implications for the workforce (Lacity and Hirschheim, 1993b; Baldwin *et al.*, 2001; Wholey *et al.*, 2001). For studies that investigate the outsourcing of multiple functions and that distinguish their results according to outsourced functions, we referred to the subsample and results of the outsourced function that is similar to CC (e.g., we used the subsample of 'system & data center operations' in Dibbern and Heinzl (2009)).

External services

This research focuses on services delivered by external providers. We excluded organizational (internal) activities related to data center virtualization and IT services provided by the internal IT department, as the managerial implications of internal IT service provision differ from those of outsourcing. For instance, internally delivered cloud services are easier to align with security, compliance, legal, and regulatory requirements than externally delivered cloud services (European Network and Information Security Agency, 2011). Thus, inhibitors such as data location (Browning and MacDonald, 2011), vendor lock-in (Armbrust *et al.*, 2010), and loss of control (Spink, 2010) play a subordinate role in internal cloud service provision.

Private sector

Because certain factors play a significant role in public sector sourcing decisions but not in private sector sourcing decisions (Arlbjørn and Freytag, 2012), we included only articles that investigate sourcing decisions for private sector organizations. For instance, only cloud service providers that are certified by the Federal Risk and Authorization Management Program (FedRAMP) are allowed to provide cloud services for US public sector organizations (General Services Administration, 2012). Furthermore, specific laws and regulations apply (e.g., the Federal Information Security Management Act of 2002; Title 44 US Code § 3541, et seq.), and different stakeholders or forces have a voice in public sector sourcing decisions (e.g., the government). Thus, to retain focus on a similar set of outsourcing decisions, we excluded public sector studies (e.g., Janssen and Joha, 2011).

Identification of 88 empirical articles on IT sourcing decisions

Execution of the search string and application of the defined inclusion criteria resulted in 48 relevant articles. Backward search (14 relevant articles) and forward search (14 relevant articles) were conducted subsequently. Additionally, we conducted a backward search on selected CC and ITO literature reviews (Lacity *et al.*, 2009; Martens and Teuteberg, 2009; Lacity *et al.*, 2010, 2011; Venters and Whitley, 2012; Yang and Tate, 2012) and identified 12 additional relevant articles. Thus, the total set of articles considered in this literature review includes 88 articles that have been published before April 2014

without limiting the time frame (see Appendix C for a list of all articles). In total, 60 articles (68%) are quantitative, and 28 are qualitative (32%). The majority of the articles originate from the ITO research stream (45 articles, 51%). ASP is the focus of 20 articles (23%), and CC is the focus of 23 articles (26%), 12 of which are dedicated to SaaS and 11 of which are dedicated to general CC.

Variable coding

We coded the 88 relevant articles based on the method developed by Jeyaraj *et al.* (2006) and applied to the ITO context by Lacity *et al.* (2010, 2011). The applied method enabled us to include results from both qualitative and quantitative studies. To aggregate the findings across studies, we adapted the iterative method of Lacity *et al.* (2010) that requires individual articles to be coded multiple times. In each iteration, we followed a three-step approach to examine a set of 20 randomly selected articles. To ensure consistent coding, two researchers independently coded all of the articles and discussed any conflicting results after each iteration.

Step 1: Code author variables

For each independent and dependent variable in an article, we coded the name and definition as given by the authors ('author variable' and 'author variable definition').

Step 2: Code relationships

We coded each examined relationship between the independent and dependent variables. To uniformly code the relationships between the independent and dependent variables from qualitative and quantitative studies, we applied the coding template of Lacity *et al.* (2010), which is depicted in Table 2. For each relationship, one of four possible values was assigned (+, −, M, 0).

Step 3: Code master variables

We independently combined and aggregated the coded author variables (see step 1) and developed a set of 'master variables' and 'master variable definitions' (see Appendix E for the variable definitions). Each researcher independently mapped the author variables of the 20 articles on the master variables. The master variables were complemented by new master variables in each iteration. If new variables were added, then previously coded articles were re-examined to determine whether any variables needed to be refined based on the addition of the new master variables. After each iteration, we met and discussed the master variables and relationship coding. Variable definitions as well as conflicting mappings and relationships were discussed and adapted. This process continued until all of the articles were coded.

Verification of coding and grouping of master variables

One researcher conducted a final check through all of the articles and coded relationships to ensure that all of the variables were coded consistently with the master variable list. For consistency checks, each master variable definition was cross-checked with the definitions of all of the author variables mapped on the master variable as well as the definitions from seminal theoretical articles (e.g., Williamson (1981) for asset specificity).

To facilitate discussion of a large number of variables that cover diverse facets of IT sourcing, we categorized the variables into seven broader categories. We divided the factors into the following categories: asset characteristics, technology characteristics, solution characteristics, client firm characteristics, individual characteristics of the decision maker, environmental characteristics, and vendor firm characteristics. The variables and categories evolved inductively from the literature.

Identification of determinant factors of cloud-sourcing decisions

To identify the determinant factors of cloud-sourcing decisions with consistent results in previous research, we counted the number of times that the relationship between a master variable and the dependent meta-variable IT sourcing decision was studied and the number of times that this relationship is reported to be positively significant, negatively significant, nonsignificant, or significant but non-directional (e.g., categorical variables, see Table 2). Next, we applied the decision rule of Lacity *et al.* (2010) and identified master variables that have been *examined multiple times* and that have produced *consistent results*. In terms of *multiple examinations*, we extracted all of the master variables that were empirically examined to influence sourcing decisions at least five times. In terms of *consistent results*, we selected only master variables for which at least 60% of the findings are consistent (i.e., significantly positive, significantly negative, or significant but non-directional). This minimum threshold ensures that more than half of the evidence produced the same findings.

We then divided our coding results into two batches. The first batch consists of coded relationships between the master variables and sourcing decisions in the ITO context, and the second batch consists of coded relationships between the

Table 2 Relationship coding scheme, adapted from Lacity *et al.* (2010)

Code	*Meaning*
+	Positive relationship: a higher value of the independent variable is associated with a higher value of the dependent variable; $P<0.05$ for quantitative studies or strong argument by authors for qualitative studies
−	Negative relationship: a higher value of the independent variable is associated with a lower value of the dependent variable; $P<0.05$ for quantitative studies or strong argument by authors for qualitative studies
M	The relationship matters: the relationship between a categorical independent variable (e.g., industry) and a dependent variable is significant but non-directional; $P<0.05$ for quantitative studies or strong argument by authors for qualitative studies
0	Relationship was studied and no significant relationship was found

master variables and sourcing decisions in the CC context (including ASP because we treated ASP as form of SaaS; see background section). We assessed each extracted master variable by comparing and contrasting the coded relationships for CC and ITO. The next section reports the results of our assessment and presents the factors with consistent empirical evidence for CC, the factors with consistent empirical evidence for ITO but only limited evidence for CC, and the factors with contradicting or inconclusive findings in the CC and ITO contexts.

Findings

The coding of 88 empirical articles on IT sourcing decisions resulted in 625 relationships between the independent and dependent variables, of which 542 have a direct influence and 83 have an indirect influence (e.g., second-order constructs or moderators) on IT sourcing decisions. Of the 542 relationships related to sourcing decisions, 272 were coded for the CC model, and 270 were coded for the ITO model. We aggregated the independent variables for all studies into a set of 111 independent master variables (see Appendix E for the definitions). The entire coding scheme is available from the authors upon request.

Appendix D lists the set of 111 master variables and the 542 aggregated relationships between the master variables and sourcing decisions. Appendix D provides details on the most frequently studied variables in the CC and ITO contexts as well as results from empirical examinations of the relationships between the independent variables and IT sourcing decisions. The findings are briefly discussed below.

Asset characteristics

We denote the asset as an object of the sourcing decision, that is, the object being outsourced. Asset characteristics are specific to the type of asset that is considered for sourcing from an external service provider, such as the strategic importance of the application being outsourced. Independent variables that examine the influence of asset characteristics on sourcing decisions are the most frequently studied among the seven categories. This category contains 17 independent variables that have been examined 140 times. Cost savings is the most commonly studied determinant (51 times: 22 times for cost savings in general, 20 times for production costs, and 9 times for transaction costs), followed by asset specificity (37 times: 15 times for technical specificity, 11 times for site specificity, and 11 times for human asset specificity), and the strategic importance of the asset (13 times).

The results for only two variables (cost savings and the strategic importance of the asset) are consistent for both CC and ITO (i.e., the same findings at least 60% of the time and at least five examinations for both CC and ITO). The results for the two variables measurement problems and technical complexity are consistent for ITO only, as research on this variable in the CC context is lacking. Further, the results for cost uncertainties are consistent in the overall sample, but research on this variable is lacking in both subsamples (investigated fewer than five times in each subsample). Finally, the results for asset specificity are consistent for CC but inconsistent for ITO.

Solution characteristics

Solution characteristics denote characteristics that are specific to a concrete solution (i.e., a cloud service), such as functional characteristics or contract specifics. The solution characteristics category consists of seven independent variables whose influence on sourcing decisions has been examined only eight times. Consequently, because of a lack of empirical evidence, none of the variables meets our inclusion criteria. The only independent variable whose influence on sourcing decisions was studied twice is perceived contract clarity, and both studies report a positive, significant relationship between perceived contract clarity and sourcing decisions (Pinnington and Woolcock, 1995; Currie *et al.*, 2004).

Technology characteristics

Technology characteristics include determinant factors that are inherent to the particular sourcing option (i.e., CC or ITO), such as the risk of losing access to data and the benefits of increased scalability. Researchers have investigated a rich array of technology characteristics as determinant factors of IT sourcing decisions, predominantly concerning the risks or benefits of the desired sourcing option. In all, the relationship between technology characteristics and sourcing decisions has been examined 128 times. Technology characteristics include 19 independent variables, nine of which are specific benefits and four of which are specific risks of the examined sourcing option. Some studies aggregate several risks or benefits within one generic risk/benefit construct (e.g., Daylami *et al.*, 2005), others investigate how specific risks/benefits influence sourcing decisions (e.g., Saya *et al.*, 2010), and others employ a two-stage model (e.g., Benlian and Hess, 2011). Some technology characteristics are conceptualized as benefits (e.g., better security (Gupta *et al.*, 2013) or increased availability (Currie *et al.*, 2004)) as well as risks (e.g., availability risks (Lechesa *et al.*, 2011) or security risks (Wu *et al.*, 2011)). The most frequently examined benefits are access to specialized resources (20 times), a focus on core competencies (17 times), and flexibility (16 times). The most frequently examined risks are security concerns (15 times), loss of control (eight times), and availability concerns (five times).

The results for only two independent variables are consistent for CC and ITO decisions (access to specialized resources and flexibility gains). The result for security risks, availability risks, reduced time to market, and perceived complexity are consistent for CC, but research on these variables in the ITO context is lacking. Focus on core competencies and quality improvements have consistent results in the ITO context; however, research on these variables in the CC context is lacking. The results for risk of losing control are consistent in the overall sample, but research on this variable is lacking in both subsamples.

Client firm characteristics

With 24 independent variables, the client characteristics category contains the most variables among the seven categories. Researchers have examined the influence of client characteristics on sourcing decisions 127 times. Client size is the most frequently discussed variable in this category (26 times). Furthermore, researchers have investigated determinant factors such as internal IT capabilities (22 times) and industry (10 times). Only the results for industry are

consistent results with both models. The results for top management support are consistent for ITO only, as findings on this variable in the CC context are scarce. Further, the results for internal IT capabilities are consistent for ITO but inconsistent for CC, and the results for client size are inconsistent with both models. The results for strategic vulnerability are consistent in the overall sample, but research on this variable is lacking in both subsamples.

Individual characteristics

Findings on the individual level are scarce, as most articles adopt a firm-level perspective. This category contains 13 variables, which researchers examined 22 times in relation to sourcing decisions. Only the influence of a decision maker's attitude toward outsourcing has been examined more than five times, and the results are consistent in the overall sample, but research on this variable is lacking in both subsamples.

Environmental characteristics

Overall, the 23 variables within the category of environmental characteristics have been examined 86 times. In our sample, uncertainty (24 times: 17 times for environmental uncertainty and seven times for behavioral uncertainty), market maturity (16 times), and institutional influences (24 times: 14 times for mimetic pressures, six times for coercive pressures, three times for normative pressures, and one time for an aggregated construct) are the most commonly examined independent variables. Market maturity is the only independent variable with consistent results across both models. The results for uncertainty are inconsistent for both CC and ITO, while the results for institutional influences are consistent for ITO but nonsignificant for CC.

Vendor firm characteristics

Vendor characteristics include eight variables that have been examined 31 times in total. Vendor service capability (14 times), support (four times), and trustworthiness (three times) are the most commonly examined independent variables in this category. Service capability is the only independent variable that has been examined more than five times, and the results are consistent for CC, although research on this variable in the ITO context is lacking.

Intermediate summary of findings

A comparison of findings from the ITO literature with findings from the CC literature, as summarized in the preceding section, revealed three types of determinant factors: (i) factors with consistent empirical evidence regarding their influence on cloud-sourcing decisions; (ii) factors with consistent empirical evidence regarding their influence on IT sourcing decisions but only limited evidence for the CC context (examined fewer than five times in the CC context); and (iii) factors with inconclusive empirical evidence in the CC and ITO contexts. For the first type of factors, the empirical results are consistent in the CC literature and are lacking in the ITO literature or are consistent in both the ITO literature and the CC literature. For the second type of factors, consistent empirical results have not yet been reported in the CC literature, but they show promise as determinant factors of cloud-sourcing decisions because the results for these factors are consistent in the overall sample and/or in the ITO sample. The third type of factors requires further discussion because the results for the two subsamples are inconclusive. Since the results for factors of type (i) and (ii) are already consistent in the literature (for either CC or ITO or for both), we do not further discuss these factors. We include these factors in our suggestions for further research, as researchers have already provided strong empirical evidence for their influence on IT sourcing decisions. Figure 1 depicts the derived factors of type (i) and (ii). To identify the most robust findings, we adapt a tiered legend from Lacity *et al.* (2010), which is based on the proportion of consistent findings in the overall sample. We use the symbol '++' if more than 80% of the evidence shows a positive and significant relationship between the independent variable and the dependent variable (IT sourcing decision). We use the symbol '+' if 60%–80% of the evidence is

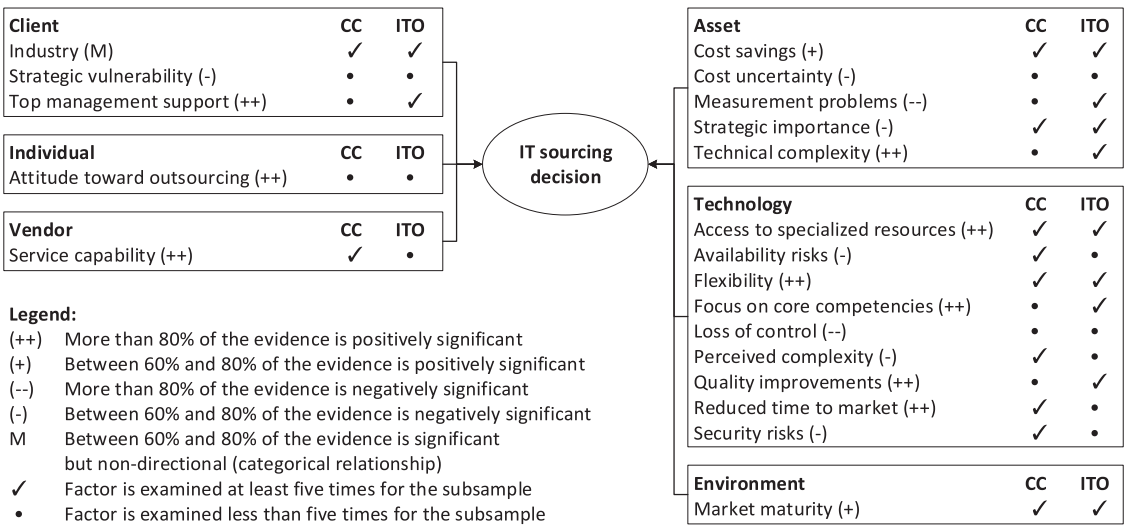


Figure 1 Determinant factors with consistent empirical evidence regarding the influence on IT sourcing decisions.

significantly positive. Likewise, we use '––' for relationships that are shown to be significantly negative more than 80% of the time and '–' if 60%–80% of the evidence is significantly negative. The symbol '✓' indicates for each subsample (CC, ITO) whether a factor is examined at least five times with consistent evidence, while the symbol '•' indicates that the factor lacks examinations for the subsample (less than five times examined).

To fulfill the objective of this article and to maintain a concise and meaningful discussion, the next section focuses on the puzzling differences and inconclusive findings in the determinant factors of sourcing decisions between CC and ITO. Table 3 summarizes the factors with inconclusive results.

Discussion of findings

In this section, we discuss the factors with inconclusive and contradicting results in the CC and ITO contexts. We therefore use the arguments in Table 1 (see background section) to examine whether the inconclusive findings can be explained by peculiarities in the CC sourcing model. We discuss each factor according to the following structure. First, we summarize the main results of the articles that examined each factor. We then discuss why there may be inconsistencies between the CC and ITO contexts and offer recommendations for future research based on our discussion. Table 4 summarizes the factors that we discuss in this section, the arguments regarding which peculiarities of CC (see Table 1) could explain the identified inconsistencies, and the factors that might induce context-specific results because of interaction effects with other factors.

Asset characteristics

Asset specificity is used in reference to three major categories of assets: site specificity, physical asset specificity (also referred to as technical specificity), and human asset specificity (Williamson, 1981).

Site specificity is examined 11 times in total, including eight times for CC and three times for ITO. Except for one occurrence for CC (Loebbecke and Huyskens, 2006), all findings show a significant, negative influence of this factor on sourcing decisions (91%) (e.g., Dibbern *et al.*, 2003). Hence, the findings are consistently negative for CC and for the overall sample, but there is a lack of evidence regarding this factor in the ITO context.

The increased network dependence of CC increases the risks that affect site specificity, such as the risk of service breakdowns because of possible network outages, which may result in a temporary loss of access to data. Hence, site specificity requires particular consideration for assets that are required on a daily basis (e.g., a customer relationship management system). Furthermore, resource pooling increases the risk of information leakage (Brender and Markov, 2013) resulting from malicious behavior in shared environments (Subashini and Kavitha, 2011). In this context, site specificity concerns not just specific resources that are only available on-site (e.g., geographical location of an investment) but also restrictions on the data that are stored and processed within the asset (Rieger *et al.*, 2013). Hence, site specificity requires particular consideration for assets that store sensitive data or that are liable to specific regulatory requirements (e.g., accounting systems). Site specificity may also be moderated by differences between service models, for instance, an asset that bears a strong competitive advantage, such as an advanced algorithm or a newly developed method (e.g., 'Summly' (Lessin, 2013)). A company may not want to outsource software development to an external contractor because knowledge could be leaked (high site specificity). However, running the algorithm in compiled code with external IaaS or PaaS solutions to increase the scalability of its use could be a valid sourcing option. By contrast, if a strategic advantage does not evolve from the asset to be outsourced (e.g., an application) but rather emerges from the data that are stored within the application (e.g., highly detailed consumer data that are stored in a market researcher's database), then cloud sourcing may not be a feasible option. Companies that rely on their unique data cannot allow such data to leak. However, outsourcing the software development of the application that grants access to their database may be an acceptable solution for such companies.

As discussed above, several indicators facilitate the need for context-specific considerations of site specificity as a determinant of cloud-sourcing decisions. In particular, we emphasize the need to distinguish the influence of site specificity according to the type of asset that is outsourced and the service model that is under consideration.

The relationship between *technical specificity* and sourcing decisions is examined 15 times in the examined articles, including 10 times in the CC context and five times in the ITO context. For ITO, the results for technical specificity are inconsistent (a significantly negative influence two of five times (40%) (Aubert *et al.*, 1996; Wholey *et al.*, 2001) and a significantly positive influence one of five times (20%) (Aubert *et al.*, 2004). For CC, six of the ten studies (60%) are consistent and report a significant, negative influence (e.g., Watjatrakul, 2005). However, 20% (two of ten times) report a significant, positive relationship (Diana, 2009; Asatiani *et al.*, 2014), which is contrary to the predictions of TCE, and an additional 20% report no significant relationship (Loebbecke and Huyskens, 2006; Benlian *et al.*, 2009).

Loebbecke and Huyskens (2006) argue that technical specificity may nevertheless generate transaction costs when applications are run remotely, but these increased transaction costs do not exceed cost savings arising from economies of scale. Thus, these cost savings may simply outweigh issues related to technical specificity. Furthermore, the influence of technical specificity is expected to vary with company size (Benlian, 2009; Asatiani *et al.*, 2014). For instance, Benlian *et al.* (2009) find that technical specificity has the strongest effect on large enterprises because large enterprises tend to have complex, highly specific business processes in place that are supported by fragmented IT infrastructures with legacy systems and

Table 3 Determinant factors with inconclusive empirical evidence regarding the influence on ITO decisions and the influence on cloud-sourcing decisions

Determinant factor	*Cloud computing*	*IT outsourcing*
Asset specificity	–	Inconsistent
Client firm internal IT capabilities	Inconsistent	––
Client firm size	Inconsistent	Inconsistent
Uncertainty	Inconsistent	Inconsistent
Institutional influences	Nonsignificant	+

Table 4 Summary of discussion of inconclusive findings, the peculiarities of CC that might explain the inconclusive findings, and recommendations for future research

Determinant factor	*Peculiarities of cloud computing that might explain inconclusive findings*	*Factors to consider in future research that might yield context-specific results*
Asset specificity		
Site specificity	• Broad network access: increased network dependence • Resource pooling: multi-tenant applications • Governance mode: different service models	• Type of asset to be outsourced (usage frequency, strategic importance) • Service model under consideration (SaaS *vs* IaaS/PaaS)
Technical specificity	• Scope: standardized services • Governance mode: different service models	• Company size • Service model under consideration (SaaS *vs* IaaS/PaaS) • Type of asset to be outsourced (strategic importance) • Environmental uncertainty
Human asset specificity	• Contract: scalability • Governance mode: different service models	• Type of asset to be outsourced (technical complexity, strategic importance) • Service model under consideration: human asset specificity as a driver for IaaS/PaaS and inhibitor for SaaS
Client – firm internal IT capabilities	• Governance mode: different service models • Scope: role of the IT department as service integrator	• Service model under consideration: lack of internal IT capabilities as driver for SaaS and inhibitor for IaaS/PaaS • Recent recruiting of IT personnel
Client firm size	• Decision process: self-service procurement	• Type of asset to be outsourced (green field or replacement of existing system)
Uncertainty		
Environmental uncertainty	• Contract: scalability, pay per use • Decision process: self-service procurement	• High demand uncertainty as driver; high product uncertainty as inhibitor • Type of asset to be outsourced (strategic importance, technical specificity)
Behavioral uncertainty	• Contract: short-term contracts, pay per use, measured service • Environment: volatile and immature market	• Type of asset to be outsourced (technical specificity, switching cost, measurement problems, strategic importance) • Market maturity
Institutional influences		
Coercive pressures	• Environment: uncertain legal conditions	• Industry • Type of asset to be outsourced (site specificity)
Mimetic pressures	• Environment: volatile and immature market	
Normative pressures	• Environment: Standardization movement • Governance mode: community cloud as deployment model	

heterogeneous applications. Thus, the role of technical specificity may vary with company size when applications or infrastructure components are outsourced.

Increasing standardization efforts in the field of CC (Bernnat *et al.*, 2012), open interfaces, interoperability, and custom-built mash-ups of cloud services enable customers to build highly specific solutions that may reduce the negative influence of technical specificity in the CC context. The market for SaaS is growing, and external business applications are increasingly provided externally (Winkler and Brown, 2013). Organizations are able to use cloud services even for complex enterprise software, such as enterprise resource planning or customer relationship management. Diana (2009) argue that highly complex IT environments may present more opportunities for outsourcing arrangements (e.g., outsourcing single, highly specific applications to an

Coercive pressures are studied six times in the examined articles, four of which are in the CC context (only one time (25%) with a significantly negative influence) and two of which are in the ITO context (both finding a significantly negative influence).

The CC environment is characterized by uncertainty and a lack of transparency that inhibits adoption, as well as an immature legal situation, as indicated by legal conflicts between the United States and Europe in data protection principles (Boehler and Ramos, 2014) or the latest information revealed about the NSA PRISM program (Cloud Security Alliance, 2013). Six studies investigate coercive pressures. The three studies that show a significant, negative influence are all in industries with strong governmental regulation, such as financial services (Ang and Cummings, 1997; Rieger *et al.*, 2013) and health care (Wholey *et al.*, 2001). Lian *et al.* (2014) investigate coercive pressures in the health-care sector (Taiwan hospitals) and find no significant influence. The authors argue that the Taiwan government has implemented an electronic medical record policy that is a driver of CC adoption rather than an inhibiting factor. The other studies showing a nonsignificant influence of coercive pressures focus on multiple industries (Borgman *et al.*, 2013; Kung *et al.*, 2013).

Hence, we argue that industry-specific effects may lead to inconsistent results regarding coercive pressures and that coercive pressures should be specifically examined with regard to companies' originating industry. In this context, interaction effects with the degree of site specificity of the outsourced asset (e.g., related to the type of data that are stored in the outsourced asset) may arise, and these effects should thus be considered in future research as well.

The results on the influence of *mimetic pressures* on sourcing decisions are inconsistent in both CC and ITO research. In total, mimetic pressures are studied 14 times, including four times in the CC context and ten times in the ITO context. The influence of mimetic pressures on cloud-sourcing decisions is significantly positive in only one of four (25%) articles (Wu *et al.*, 2011), whereas in eight of ten (80%) articles, the influence of mimetic pressures on ITO decisions is positive and significant (e.g., Fiedler *et al.*, 2013).

The findings regarding the effect of mimetic pressures on cloud-sourcing decisions may be inconclusive because mimetic pressures may influence sourcing decisions indirectly rather than directly. Saya *et al.* (2010) report that institutional influences significantly affect perceived risks (e.g., security) and perceived benefits (e.g., accessibility) but do not directly affect cloud-sourcing decisions. Similarly, Benlian *et al.* (2009) find that the opinions of experts or market researchers play a major role in shaping the attitudes of top management toward cloud sourcing but that they have no direct influence on sourcing decisions. These authors argue that organizations appear 'not to blindly follow the recommendations of other organizations by unreflectively imitating their behavior. Instead, the opinions of third parties seem to inform in the IT user companies' process of building their own attitude about SaaS'. This argument is consistent with the nonsignificant results of Blaskovich and Mintchik (2010), who find a significantly positive influence of mimetic pressures on cloud-sourcing decisions only when CIO skills are weak. Considering the high level of uncertainty within the CC environment, mimicking organizations that are perceived to be more legitimate or successful is a response to uncertainty (DiMaggio and Powell, 1983). Decision makers are influenced by the environment both inside and outside their organization, and both sources of influence are strong predictors of sourcing decisions (Loh and Venkatraman, 1992b; Dibbern *et al.*, 2012). Hu *et al.* (1997) argue that the combined effects of external media, vendor pressure, and internal communications among managers at the personal level significantly influence sourcing decisions. More specifically, Morgan and Conboy (2013) report that customers are negatively preoccupied about the term cloud, and because of this perception, service providers avoid mentioning the term cloud *per se* and instead refer to a new service delivery model.

Although findings on the direct influence of mimetic pressures on cloud-sourcing decisions are scarce in the CC context, we argue that mimetic influences should be considered in further research as a determinant factor of cloud-sourcing decisions, particularly with respect to other factors from social or organizational theories, as research has shown that these relationships are strong and significant in the CC context (Benlian *et al.*, 2009; Saya *et al.*, 2010; Wu *et al.*, 2011).

Normative pressures are investigated only three times (including two times in the CC context). The results for CC show a positive, significant influence in one of the two studies (50%) (Kung *et al.*, 2013), and the single study in the ITO context (Fiedler *et al.*, 2013) also shows a positive, significant relationship.

Because of the limited number of empirical investigations on normative pressures, we cannot draw conclusions based on our coding results. However, with the maturation of the cloud market and the establishment of cloud-dedicated industry associations, such as the Cloud Security Alliance or EuroCloud, a large number of norms and best practices for CC have been established. The International Organization for Standardization (ISO) is currently developing a code of practice for information security controls for CC in alignment with the prominent ISO 27000 series of standards (International Organization for Standardization, 2013). Furthermore, community cloud as a deployment model enables industry associations to provide members with cloud services that implement industry best practices and that are compliant with industry-specific regulations.

Increasing standardization efforts that are particularly suited to CC and the emergence of community cloud platforms for specific industries might drive organizations to consider CC adoption. We therefore suggest that future research incorporate normative pressures and challenge the argument regarding whether cloud-specific standardization movements and community cloud platforms drive cloud-sourcing decisions.

Implications for research

The synthesis of the literature on determinant factors of IT sourcing decisions enables us to draw conclusions based on not only the in-depth discussion of contradicting findings between research on CC and research on ITO (Figure 1, Table 3, and Table 4) but also the emergent trends in the literature. This section discusses implications for future research based on both types of findings.

To answer our first research question, we examined what we can learn from the rich body of research on ITO to identify

result in common challenges (e.g., internet connectivity, monitoring and pricing, resource management) and similar factors driving sourcing decisions. However, given our discussion (e.g., regarding the service model-specific considerations of asset specificity or the internal IT capabilities of the client firm) and empirical evidence related to outsourcing different types of IS functions (Lacity and Hirschheim, 1993a; Benlian *et al.*, 2009; Dibbern and Heinzl, 2009), the determinant factors of decisions to source SaaS solutions may considerably differ from the determinant factors of decisions to source IaaS or PaaS solutions. Although research on IaaS and PaaS is evolving (e.g., Giessmann and Stanoevska, 2012; Repschlaeger *et al.*, 2012), research on the determinant factors of sourcing decisions related to these service models and the differences between the service models is currently lacking. We therefore call for further research on specific cloud service models, particularly research on IaaS and PaaS adoption and comparative research on the service models.

Additionally, we encourage further research to provide inter-group comparisons, as extant research has already revealed various differences between investigated groups, such as adopters *vs* non-adopters (Benlian and Hess, 2011; Borgman *et al.*, 2013; Lian *et al.*, 2014), and differences among countries (Apte *et al.*, 1997), industry sectors (Hancox and Hackney, 2000; Currie *et al.*, 2004), and outsourced functions (Benlian *et al.*, 2009; Dibbern and Heinzl, 2009). For instance, Benlian and Hess (2011) find noteworthy differences between the influences of benefits and risks for adopters *vs* non-adopters of SaaS. IT executives in non-adopter firms expect SaaS to improve the quality of their services and to enable a focus on core competencies, but these factors fail to drive SaaS adoption at adopter firms. In addition to identifying these differences, future research should provide evidence for the underlying mechanisms driving differences, for instance, between adopter and non-adopter firms.

Drawing from the entire set of 88 articles, we find that researchers have applied various theories to examine IT sourcing decisions. Overall, TCE (32 times) and the resource-based view of the firm (16 times) are the most frequently used theories. The majority of research applies macro-perspective theories, for both CC and ITO. Macro-perspective theories consider factors from the standpoint of an entire organization, whereas a micro-level perspective concerns an individual decision makers' standpoint (Vetter *et al.*, 2011). In all, 80 articles apply a macro-perspective (e.g., Yao *et al.*, 2010), only five articles investigate determinant factors on the micro level (e.g., Li *et al.*, 2006), and only three adopt a mixed perspective (Dibbern *et al.*, 2003; Benlian *et al.*, 2009; Lian *et al.*, 2014). Researchers applying multiple perspectives or integrating micro- and macro-level theories emphasize the strong explanatory power of behavioral theories and demand further combinations of theories from various fields, such as economics and psychology. We therefore emphasize the need for further research applying a micro-level perspective to investigate cloud-sourcing decisions. We suggest that future research combine factors from strategic and economic theories with factors from social or organizational theories, particularly determinant factors from the categories individual characteristics and environmental characteristics.

Most of the studies (63 articles, 72%) draw empirical data from top management (executive board or senior managers) and neglect other members of the procurement team. However, organizational decision making involves multiple stakeholders from inside and outside the organization (Heckman, 2003; Verville and Halingten, 2003). Each stakeholder group shapes the decision process and outcome; engages in different activities, possesses specific decision rights, responsibilities, and information needs; and pursues a unique set of goals (Webster and Wind, 1972; Howcroft and Light, 2010; Bidwell, 2012; Harnisch *et al.*, 2013; Winkler and Brown, 2013). For instance, a legal department may require different information and may have different requirements for cloud services than an IT department or the business unit (Schneider *et al.*, 2014). In particular, because of the peculiarities of CC, such as the changed role of the IT department and the adapted decision processes, research adopting the perspectives of multiple stakeholders regarding cloud-sourcing decisions might yield fruitful results. None of the articles investigated in this study adopts a multi-level perspective and evaluates differences in determinant factors between stakeholder groups or between hierarchical levels in a company. We therefore propose that future research adopt a multi-stakeholder perspective and evaluate differences in stakeholder perceptions. This suggestion is consistent with recent calls for research on CC, ITO, and packaged software acquisition (Benlian *et al.*, 2009; Howcroft and Light, 2010; Bidwell, 2012).

Table 5 summarizes our discussion and proposes future research directions.

Limitations

This research has certain limitations. First, we conducted a systematic literature review to derive the determinant factors of cloud-sourcing decisions. Our results inform a set of determinant factors to provide a basis for further research on cloud-sourcing decisions. Although our derived set of determinant factors does not depict the interdependencies, mediation, or moderation effects of factors, we provide suggestions regarding the moderating or interaction effects among the factors in our discussion (see Table 4 for a summary). Second, the process of selecting studies may also generate discussion. For articles focusing on ITO, authors do not always specify the type of ITO decision on which the article focuses. Some studies examine mixed types of outsourcing activities, for instance, by combining software development outsourcing and datacenter operations in a single sample, or do not clearly define the type of outsourcing that they investigate. However, by having two researchers review each article, we aimed to reduce the subjectivity of the article selection. Third, the body of knowledge considered for this research is limited to the keyword search within the top 50 publications of the AIS journal ranking and selected conferences. By applying forward and backward search processes, we aimed to mitigate this limitation. Although we cannot guarantee that we identified every CC and ITO article that investigates sourcing decisions, we are confident that we achieved good and reasonable coverage, which is preferred to 'a comprehensive one that would make a review process at best ephemeral if not unachievable' (Rowe, 2014: 246). Finally, we acknowledge that our coding may be subjective to a certain extent. In particular, this subjectivity may arise if researchers do not clearly define the variables that are used or if the definitions of the author variables to map on a master variable are ambiguous.

Table 5 Proposed directions for future research

- Incorporate determinant factors from Figure 1 in future research
  - Consider factors from the non-technology context for cloud-sourcing decisions
  - Consider 'good' predictors of cloud-sourcing decisions in the ITO context
  - Consider 'good' predictors of ITO decisions in the CC context
- Empirically clarify the reasons for the inconclusive findings between the determinant factors of cloud-sourcing decisions and the determinant factors of ITO decisions and incorporate the suggested factors into context-specific future research (see Table 4)
  - Conduct in-depth research (e.g., field or case studies) to identify how and why differences manifest between CC and ITO (e.g., decision processes)
  - Conduct inter-group comparative research to clarify context-specific deviations
  - Conduct service-model specific research to identify differences in the determinant factors between service models (particularly IaaS and PaaS)
- Address gaps identified in research on CC and ITO decision making
  - Apply micro-level theories and focus on individual-level sourcing decisions
  - Apply multi-stakeholder perspectives to evaluate differences in stakeholder perceptions regarding determinant factors

However, we are confident about the reliability and validity of our findings because of the rigorous coding approach based on independent coding by two researchers and discussion of diverging variable coding.

Conclusion

In this work, we conducted a systematic literature review, surveyed and synthesized prior empirical research concerning decision making in the CC and ITO contexts, and examined the rich body of research on ITO to identify determinant factors of sourcing decisions in the CC context. We linked the ITO literature with the CC literature and discussed what the CC literature can learn from the ITO literature (i.e., which findings are transferable and which findings might need reconsideration). We then discussed whether the peculiarities of CC can explain differences between the determinant factors of ITO decisions and the determinant factors of cloud-sourcing decisions. We extracted the most frequently studied determinant factors with robust results, discussed the inconclusive findings, identified gaps in the literature, and suggested paths for future research. Furthermore, we discussed interdependencies between the variables and peculiarities of CC that may elucidate the inconsistent findings in prior research. Therefore, the article serves as both a repository of past research and a guide for future research.

The contribution of this review is threefold. First, we derived a set of determinant factors of cloud-sourcing decisions from research within the IS community. Our results provide a basis for future research in the CC context and contribute to the development of theory in the IS domain. We thereby answer the call of Lacity *et al.* (2011) for the development of an 'endogenous ITO theory rather than continuing to rely heavily on reference discipline theories' (p. 147). IS researchers can draw from our derived set of determinant factors of cloud-sourcing decisions to advance research on decision making or acquisition in the field of CC. By discussing inconclusive findings regarding the determinant factors of cloud-sourcing decisions, we identified contextual factors that are specific to the CC sourcing model and that may alter the influence of specific determinant factors of sourcing decisions. Based on these discussions, we provided concrete paths for future research endeavors. Second, we applied a method developed in IS research (Jeyaraj *et al.*, 2006) and demonstrated its applicability to a different research setting. Third, our work contributes to practice, as the determinant factors of cloud-sourcing decisions serve as a basis for practitioner-oriented guidelines and best practices regarding how to select and offer cloud services. Furthermore, practitioners may use the set of determinant factors to guide their procurement processes and to identify challenges that may arise during the adoption, acquisition, or integration of cloud services.

Acknowledgements

The research was conducted in context of the Value4Cloud research project, funded by the German Federal Ministry for Economics and Technology (grant no. 01MD11043A). The authors would like to thank Jens Lansing and Sebastian Lins for their assistance with this research and their constructive feedback on previous drafts of this paper. We also would like to thank the Senior Editor and anonymous reviewers for their valuable comments and suggestions to improve the quality of this paper.

References

Aggarwal, R. and Singh, H. (2013). Differential Influence of Blogs across Different Stages of Decision Making: The case of venture capitalists, *MIS Quarterly* **37**(4): 1093–1112.

Akhilesh, B. (2000). A Study of Senior Information Systems Managers' Decision Models in Adopting New Computing Architectures, *Journal of the AIS* **1**(1): 1–56.

Al-Qirim, N.A. (2003). The Strategic Outsourcing Decision of IT and Ecommerce: The case of small businesses in New Zealand, *Journal of Information Technology Cases And Applications* **5**(3): 32–56.

Alaghehband, F.K., Rivard, S., Wu, S. and Goyette, S. (2011). An Assessment of the Use of Transaction Cost Theory in Information Technology Outsourcing, *The Journal of Strategic Information Systems* **20**(2): 125–138.

Ang, S. and Cummings, L.L. (1997). Strategic Response to Institutional Influences on Information Systems Outsourcing, *Organization Science* **8**(3): 235–256.

Ang, S. and Straub, D.W. (1998). Production and Transaction Economies and IS Outsourcing: A study of the U.S. banking industry, *MIS Quarterly* **22**(4): 535–552.

Apte, U.M., Sobol, M.G., Hanaoka, S., Shimada, T., Saarinen, T., Salmela, T. and Vepsalainen, A.P.J. (1997). IS Outsourcing Practices in the USA, Japan and Finland: A comparative study, *Journal of Information Technology* **12**(4): 289–304.

Arlbjørn, J.S. and Freytag, P.V. (2012). Public Procurement vs Private Purchasing: Is there any foundation for comparing and learning across the sectors? *International Journal of Public Sector Management* **25**(3): 203–220.

Dynamic Audit Services for Outsourced Storages in Clouds

Yan Zhu, Gail-Joon Ahn, *Senior Member, IEEE*, Hongxin Hu, *Student Member, IEEE*, Stephen S. Yau, *Fellow, IEEE*, Ho G. An, and Shimin Chen

Abstract—In this paper, we propose a dynamic audit service for verifying the integrity of an untrusted and outsourced storage. Our audit service is constructed based on the techniques, fragment structure, random sampling and index-hash table, supporting provable updates to outsourced data and timely anomaly detection. In addition, we propose a method based on probabilistic query and periodic verification for improving the performance of audit services. Our experimental results not only validate the effectiveness of our approaches, but also show our audit system verifies the integrity with lower computation overhead and requiring less extra storage for audit metadata.

Index Terms—Storage Security, Provable Data Possession, Audit Service, Cloud Storage.

1 Introduction

Cloud computing provides a scalable environment for growing amounts of data and processes that work on various applications and services by means of on-demand self-services. Especially, the outsourced storage in clouds has become a new profit growth point by providing a comparably low-cost, scalable, location-independent platform for managing clients' data. The cloud storage service (CSS) relieves the burden for storage management and maintenance. However, if such an important service is vulnerable to attacks or failures, it would bring irretrievable losses to the clients since their data or archives are stored in an uncertain storage pool outside the enterprises. These security risks come from the following reasons: first, the cloud infrastructures are much more powerful and reliable than personal computing devices, but they are still susceptible to internal threats (e.g., via virtual machine) and external threats (e.g., via system holes) that can damage data integrity [1]; second, for the benefits of possession, there exist various motivations for cloud service providers (CSP) to behave unfaithfully towards the cloud users [2]; furthermore, disputes occasionally suffer from the lack of trust on CSP since the data changes may not be timely known by the cloud users, even if these disputes may result from the users' own improper operations [3]. Therefore, it is necessary for cloud service providers to offer an efficient audit service to check the integrity and availability of stored data [4].

Security audit is an important solution enabling traceback and analysis of any activities including data accesses, security breaches, application activities, and so on. Data security tracking is crucial for all organizations that should comply with a wide range of federal regulations including the Sarbanes-Oxley Act, Basel II, HIPAA, and so on.[1] Furthermore, compared to the common audit, the audit services for cloud storages should provide clients with a more efficient proof for verifying the integrity of stored data. Unfortunately, the traditional cryptographic technologies, based on hash functions and signature schemes, cannot support for data integrity verification without a local copy of data. In addition, it is evidently impractical for audit services to download the whole data for checking data validation due to the communication cost, especially for large-size files. Therefore, following security and performance objectives should be addressed to achieve an efficient audit for outsourced storage in clouds:

Public auditability: to allow a third party auditor (TPA) or clients with the help of TPA to verify the correctness of cloud data on demand without retrieving a copy of the whole data or introducing additional on-line burden to cloud services;

Dynamic operations: to ensure there is no attack to compromise the security of verification protocol or cryptosystem by using dynamic data operations;

Timely detection: to detect data errors or losses in outsourced storage, as well as anomalous behaviors of data operations in a timely manner;

Effective forensic: to allow TPA to exercise strict au-

1. Source: http://www.hhs.gov/ocr/privacy/.

- *A conference paper entitled "Dynamic Audit Services for Integrity Verification of Outsourced Storages in Clouds" appeared in Proc. of the 26th Annual ACM Symposium on Applied Computing (SAC), Taiwan, 2011, pp. 1550-1556.*
- *Y. Zhu and S. Chen are with the Institute of Computer Science and Technology, Peking University, Beijing 100871, China, and the Beijing Key Laboratory of Internet Security Technology, Peking University, Beijing 100871, China. E-mail: {yan.zhu,chensm}@pku.edu.cn.*
- *G.-J. Ahn, H. Hu, S.S. Yau, and H.G. An are with the School of Computing, Informatics, and Decision Systems Engineering, Arizona State University, Tempe, Arizona, 85287. E-mail: {gahn,hxhu,yau,ho.an}@asu.edu.*

Digital Object Indentifier 10.1109/TSC.2011.51

TABLE 1
Comparison of POR/PDP schemes for a file consisting of n blocks.

Scheme	CSP comp.	Client Comp.	Comm.	Frag.	Privacy	Dynamic Operations			Prob. of Detection
						modify	insert	delete	
PDP[5]	$O(t)$	$O(t)$	$O(1)$		✓				$1-(1-\rho)^t$
SPDP[6]	$O(t)$	$O(t)$	$O(t)$	✓	✓	✓♯		✓♯	$1-(1-\rho)^{t\cdot s}$
DPDP-I[7]	$O(t\log n)$	$O(t\log n)$	$O(t\log n)$		✓	✓	✓	✓	$1-(1-\rho)^t$
DPDP-II[7]	$O(t\log n)$	$O(t\log n)$	$O(t\log n)$			✓	✓	✓	$1-(1-\rho)^{\Omega(n)}$
CPOR-I[8]	$O(t)$	$O(t)$	$O(1)$						$1-(1-\rho)^t$
CPOR-II[8]	$O(t+s)$	$O(t+s)$	$O(s)$	✓					$1-(1-\rho)^{t\cdot s}$
Our Scheme	$O(t+s)$	$O(t+s)$	$O(s)$	✓	✓	✓	✓	✓	$1-(1-\rho)^{t\cdot s}$

s is the number of sectors in each block, t is the number of sampling blocks, ♯ indicates that an operation is performed with a limited (pre-determined) number of times, ρ is the probability of block corruption in a cloud server.

dit and supervision for outsourced data, and offer efficient evidences for anomalies; and

Lightweight: to allow TPA to perform audit tasks with the minimum storage, lower communication cost and less computation overhead.

In this paper, we introduce a dynamic audit service for integrity verification of untrusted and outsourced storages. Constructed on interactive proof system (IPS) with the zero-knowledge property, our audit service can provide public auditability without downloading raw data and protect privacy of the data. Also, our audit system can support dynamic data operations and timely anomaly detection with the help of several effective techniques, such as fragment structure, random sampling, and index-hash table. We also propose an efficient approach based on probabilistic query and periodic verification for improving the performance of audit services. A proof-of-concept prototype is also implemented to evaluate the feasibility and viability of our proposed approaches. Our experimental results not only validate the effectiveness of our approaches, but also show our system does not create any significant computation cost and require less extra storage for integrity verification.

We list the features of our scheme in Table 1. We also make a comparison of related techniques, involving provable data possession (PDP) [5], scalable PDP (SPDP) [6], dynamic PDP (DPDP) [7], and compact proofs of retrievability (CPOR) [8]. It clearly shows that our scheme not only supports complete privacy protection and dynamic data operations, but also enables significant savings in computation and communication costs, as well as a high detection probability of disrupted blocks.

The rest of the paper is organized as follows. Section 2 describes the research background and related work. Section 3 addresses our audit system architecture and main techniques. Sections 4 and Section 5 describe the definition and construction of corresponding algorithms, respectively. In Sections 6, we present the security of our schemes along with the performance of experimental results in Section 7. Finally, we conclude this paper in Section 8.

2 Background and Related Work

Traditional cryptographic technologies for data integrity and availability, based on hash functions and signature schemes [9], [10], [11], cannot work on the outsourced data without a local copy of data. In addition, it is not a practical solution for data validation by downloading them due to the expensive communications, especially for large-size files. Moreover, the ability to audit the correctness of data in a cloud environment can be formidable and expensive for cloud users. Therefore, it is crucial to realize public auditability for CSS, so that data owners may resort to a third party auditor (TPA), who has expertise and capabilities that a common user does not have, for periodically auditing the outsourced data. This audit service is significantly important for digital forensics and data assurance in clouds.

To implement public auditability, the notions of proof of retrievability (POR) [2] and provable data possession (PDP) [5] have been proposed by some researchers. These approaches were based on a probabilistic proof technique for a storage provider to prove that clients' data remain intact. For ease of use, some POR/PDP schemes work on a publicly verifiable way, so that anyone can use the verification protocol to prove the availability of the stored data. Hence, they help accommodate the requirements from public auditability. POR/PDP schemes evolved around an untrusted storage offer a publicly accessible remote interface to check the tremendous amount of data.

There exist some solutions for audit services on outsourced data. For example, Xie *et al.* [12] proposed an efficient method on content comparability for outsourced database, but it was not suitable for irregular data. Wang *et al.* [13] also provided a similar architecture for public audit services. To support their architecture, a public audit scheme was proposed with privacy-preserving property. However, the lack of rigorous performance analysis for a constructed audit system greatly affects the practical application of their scheme. For instance, in this scheme an outsourced file is directly split into n blocks, and then each block generates a verification tag. In order to maintain

security, the length of block must be equal to the size of cryptosystem, that is, 160 bits which are 20 bytes. This means that 1M bytes file is split into 50,000 blocks and generates 50,000 tags [8], and the storage of tags is at least 1M bytes. Therefore, it is inefficient to build an audit system based on this scheme. To address such a problem, we introduce a fragment technique to improve the system performance and reduce the extra storage (see Section 3.1).

Another major concern is the security issue of dynamic data operations for public audit services. In clouds, one of the core design principles is to provide dynamic scalability for various applications. This means that remotely stored data might be not only accessed by the clients, but also dynamically updated by them, for instance, through block operations such as modification, deletion and insertion. However, these operations may raise security issues in most of existing schemes, e.g., the forgery of the verification metadata (called as tags) generated by data owners and the leakage of the user's secret key. Hence, it is crucial to develop a more efficient and secure mechanism for dynamic audit services, in which a potential adversary's advantage through dynamic data operations should be prohibited.

3 Architecture and Techniques

We introduce an audit system architecture for outsourced data in clouds as shown in Fig. 1. In this architecture, we consider that a data storage service involves four entities: data owner (DO), who has a large amount of data to be stored in the cloud; cloud service provider (CSP), who provides data storage service and has enough storage space and computation resources; third party auditor (TPA), who has capabilities to manage or monitor the outsourced data under the delegation of data owner; and authorized applications (AA), who have the right to access and manipulate the stored data. Finally, application users can enjoy various cloud application services via these authorized applications.

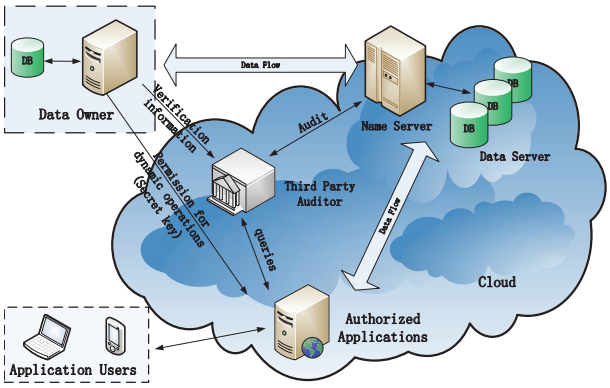


Fig. 1. Audit system architecture.

We assume the TPA is reliable and independent through the following audit functions: TPA should be able to make regular checks on the integrity and availability of the delegated data at appropriate intervals; TPA should be able to organize, manage, and maintain the outsourced data instead of data owners, and support dynamic data operations for authorized applications; and TPA should be able to take the evidences for disputes about the inconsistency of data in terms of authentic records for all data operations.

To realize these functions, our audit service is comprised of three processes:

Tag Generation: the client (data owner) uses a secret key sk to pre-process a file, which consists of a collection of n blocks, generates a set of public verification parameters (PVP) and index-hash table (IHT) that are stored in TPA, transmits the file and some verification tags to CSP, and may delete its local copy (see Fig. 2(a));

Periodic Sampling Audit: by using an interactive proof protocol of retrievability, TPA (or other applications) issues a "Random Sampling" challenge to audit the integrity and availability of the outsourced data in terms of verification information (involving PVP and IHT) stored in TPA (see Fig. 2(b)); and

Audit for Dynamic Operations: An authorized application, who holds a data owner's secret key sk, can manipulate the outsourced data and update the associated IHT stored in TPA. The privacy of sk and the checking algorithm ensure that the storage server cannot cheat the authorized applications and forge the valid audit records (see Fig. 2(c)).

In general, the authorized applications should be cloud application services inside clouds for various application purposes, but they must be specifically authorized by data owners for manipulating outsourced data. Since the acceptable operations require that the authorized applications must present authentication information for TPA, any unauthorized modifications for data will be detected in audit processes or verification processes. Based on this kind of strong authorization-verification mechanism, we assume neither CSP is trusted to guarantee the security of stored data, nor a data owner has the capability to collect the evidence of CSP's faults after errors have been found.

The ultimate goal of this audit infrastructure is to enhance the credibility of cloud storage services, but not to increase data owner's burden. Therefore, TPA should be constructed in clouds and maintained by a CSP. In order to ensure the trust and security, TPA must be secure enough to resist malicious attacks, and it should be strictly controlled to prevent unauthorized accesses even for internal members in clouds. A more practical way is that TPA in clouds should be mandated by a trusted third party (TTP). This mechanism not only improves the performance of an

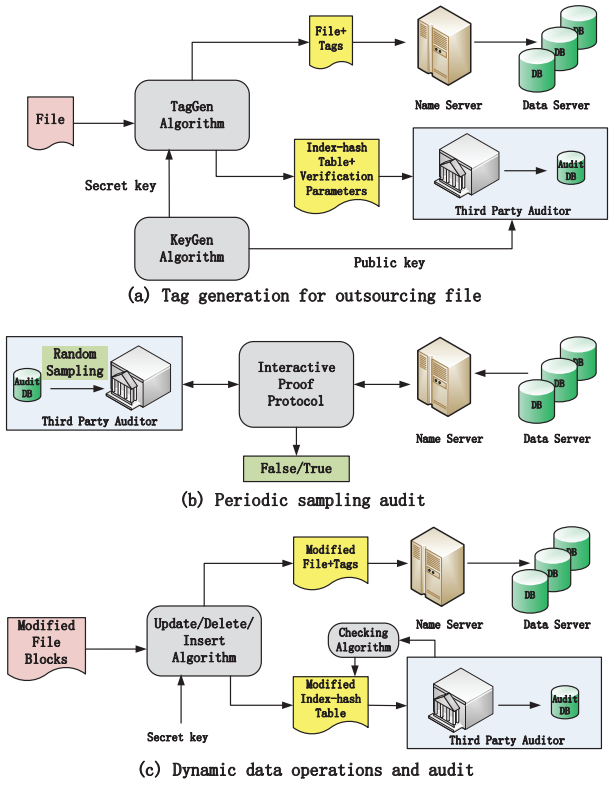


Fig. 2. Three processes of our audit system.

audit service, but also provides the data owner with a maximum access transparency. This means that data owners are entitled to utilize the audit service without additional costs.

The aforementioned processes involve some procedures: $KeyGen$, $TagGen$, $Update$, $Delete$, $Insert$ algorithms, as well as an *Interactive Proof Protocol* of Retrievability. We make use of following techniques to construct corresponding algorithms and protocols.

3.1 Fragment Structure and Secure Tags

To maximize the storage efficiency and audit performance, our audit system introduces a general fragment structure for outsourced storages. An instance for this framework which is used in our approach is shown in Fig. 3: an outsourced file F is split into n blocks $\{m_1, m_2, \cdots, m_n\}$, and each block m_i is split into s sectors $\{m_{i,1}, m_{i,2}, \cdots, m_{i,s}\}$. The fragment framework consists of n block-tag pair (m_i, σ_i), where σ_i is a signature tag of a block m_i generated by some secrets $\tau = (\tau_1, \tau_2, \cdots, \tau_s)$. We can use such tags and corresponding data to construct a response in terms of the TPA's challenges in the verification protocol, such that this response can be verified without raw data. If a tag is unforgeable by anyone except the original signer, we call it a *secure tag*.

Finally, these block-tag pairs are stored in CSP and the encrypted secrets τ (called as PVP) are in TTP. Although this fragment structure is simple and straightforward, but the file is split into $n \times s$ sectors and each block (s sectors) corresponds to a tag, so that the storage of signature tags can be reduced with the increase of s. Hence, this structure can reduce the extra storage for tags and improve the audit performance.

There exist some schemes for the convergence of s blocks to generate a secure signature tag, e.g., MAC-based, ECC or RSA schemes [5], [8]. These schemes, built from collision-resistance hash functions (see Section 5) and a random oracle model, support the features of scalability, performance and security.

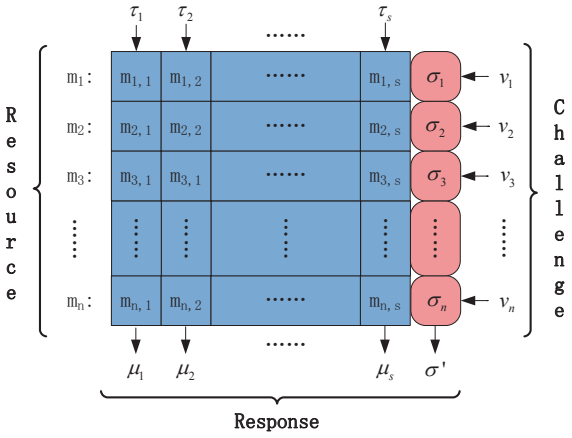


Fig. 3. Fragment structure and sampling audit.

3.2 Periodic Sampling Audit

In contrast with "whole" checking, random "sampling" checking greatly reduces the workload of audit services, while still achieves an effective detection of misbehaviors. Thus, a probabilistic audit on sampling checking is preferable to realize the anomaly detection in a timely manner, as well as to rationally allocate resources. The fragment structure shown in Fig. 3 provides probabilistic audit as well: given a randomly chosen challenge (or query) $Q = \{(i, v_i)\}_{i \in I}$, where I is a subset of the block indices and v_i is a random coefficient, an efficient algorithm is used to produce a constant-size response $(\mu_1, \mu_2, \cdots, \mu_s, \sigma')$, where μ_i comes from all $\{m_{k,i}, v_k\}_{k \in I}$ and σ' is from all $\{\sigma_k, v_k\}_{k \in I}$. Generally, this algorithm relies on homomorphic properties to aggregate data and tags into a constant-size response, which minimizes network communication costs.

Since the single sampling checking may overlook a small number of data abnormality, we propose a periodic sampling approach to audit outsourced data, which is named as *Periodic Sampling Audit*. With this approach, the audit activities are efficiently scheduled in an audit period, and a TPA merely needs to access small portions of files to perform audit in each activity. Therefore, this method can detect exceptions periodically, and reduce the sampling numbers in each audit.

3.3 Index-Hash Table

In order to support dynamic data operations, we introduce a simple index-hash table to record the changes of file blocks, as well as generate the hash value of each block in the verification process. The structure of our index-hash table is similar to that of file block allocation table in file systems. Generally, the index-hash table χ consists of serial number, block number, version number, and random integer (see Table 2 in Section 5). Note that we must assure all records in the index-hash table differ from one another to prevent the forgery of data blocks and tags. In addition to recording data changes, each record χ_i in the table is used to generate a unique hash value, which in turn is used for the construction of a signature tag σ_i by the secret key sk. The relationship between χ_i and σ_i must be cryptographically secure, and we make use of it to design our verification protocol.

Although the index-hash table may increase the complexity of an audit system, it provides a higher assurance to monitor the behavior of an untrusted CSP, as well as valuable evidence for computer forensics, due to the reason that anyone cannot forge the valid χ_i (in TPA) and σ_i (in CSP) without the secret key sk.

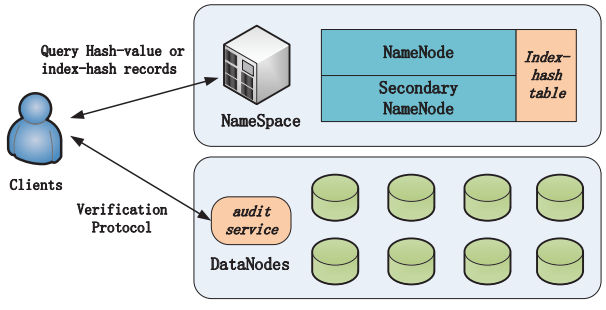


Fig. 4. An example of hash index hierarchy in Hadoop distributed file system (HDFS).

In practical applications, this architecture can be constructed into a virtualization infrastructure of cloud-based storage service [14]. In Fig. 4, we show an example of Hadoop distributed file system (HDFS) [2], which is a distributed, scalable, and portable file system [15]. HDFS' architecture is composed of NameNode and DataNode, where NameNode maps a file name to a set of indexes of blocks and DataNode indeed stores data blocks. To support dynamic audit, the index-hash table and the metadata of NameNode should be integrated together to provide an enquiry service for the hash value $\xi^{(3)}_{i,k}$ or index-hash record χ_i. Based on these hash values, the clients or TPA can implement a verification protocol via audit services. Hence, it is easy to replace the common checksum algorithm with our scheme for anomaly detection without downloading data in current HDFS.

2. Hadoop enables applications to work with thousands of nodes and petabytes of data, and it has been adopted by currently mainstream cloud platforms from Apache, Google, Yahoo, Amazon, IBM and Sun.

4 Algorithms for Audit System

In this section we describe the construction of algorithms in our audit architecture. Firstly, we present the definitions for the tag generation process as follows:

$\mathcal{K}eyGen$ (1^κ): takes a security parameter κ as an input, and returns a public/secret keypair (pk, sk); and

$\mathcal{T}agGen$ (sk, F): takes a secret key sk and a file F, and returns a triple (τ, ψ, σ), where τ denotes the secret used to generate verification tags, ψ is a set of public verification parameters u and index-hash table χ, i.e., $\psi = (u, \chi)$, and σ denotes a set of tags.

A data owner or authorized applications only need to save the secret key sk–that is, sk would not be necessary for the verification/audit process. The secret of the processed file τ can be discarded after tags are generated due to public verification parameters u.

Fig. 5 demonstrates the workflow of our audit system. Suppose a data owner wants to store a file in a storage server, and maintains a corresponding authenticated index structure at a TPA. As shown in Fig. 5(a), using $KeyGen()$, the owner firstly generates a public/secret keypair (pk, sk) by himself or the system manager, and sends his public key pk to TPA. Note that TPA cannot obtain the client's secret key sk. Then, the owner chooses a random secret τ and invokes $TagGen()$ to produce public verification information $\psi = (u, \chi)$ and signature tags σ, where τ is unique for each file and χ is an index-hash table. Finally, the owner sends ψ and (F, σ) to TPA and CSP, respectively.

4.1 Supporting Periodic Sampling Audit

At any time, TPA can check the integrity of a file F as follows: TPA first queries database to obtain the verification information ψ and initializes an interactive protocol $Proof(CSP, Client)$; then, it performs a 3-move proof protocol: *Commitment*, *Challenge*, and *Response*; and it finally verifies the interactive data to get the results. In fact, since our scheme is a publicly verifiable protocol, anyone can run this protocol, but s/he is unable to get any advantage to break the cryptosystem, even if TPA and CSP cooperate for an attack. Let $P(x)$ denotes the subject P holds the secret x and $\langle P, V\rangle(x)$ denotes both parties P and V share a common data x in a protocol. This process can be defined as follows:

$\mathcal{P}roof$ (CSP, TPA): is an interactive proof protocol between CSP and TPA, that is, $\langle CSP(F, \sigma), TPA\rangle(pk, \psi)$, where a public key pk and a set of public parameters ψ are the common inputs between TPA and CSP, and CSP takes a file F and a set of tags σ. At the end of the protocol, TPA returns $\{0|1\}$, where 1 means the file is correctly stored on the server.

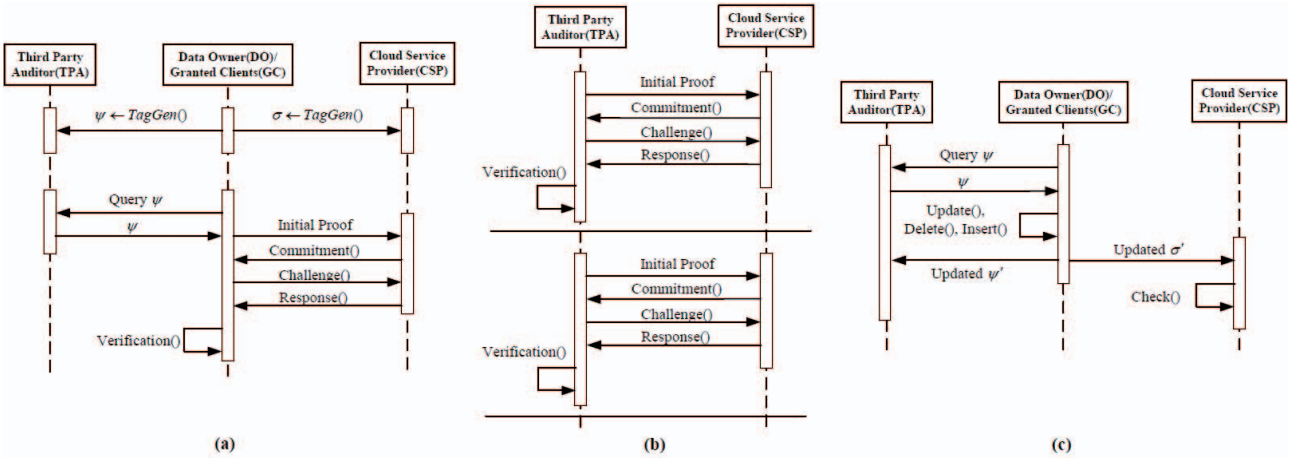


Fig. 5. Workflow of audit system: (a) tag generation and user's verification, (b) periodic sampling audit, and (c) dynamic data operations and audit.

An audit service executes the verification process periodically by using the above-mentioned protocol. Fig. 5(b) shows such a two-party protocol between TPA and CSP, i.e., $Proof(CSP, TPA)$, without the involvement of a client (DO or AA). It also shows two verification processes. To improve the efficiency of verification process, TPA performs audit tasks based on a probabilistic sampling.

4.2 Supporting Dynamic Data Operations

In order to meet the requirements from dynamic scenarios, we introduce following definitions for dynamic data operations:

$\mathcal{U}pdate(sk, \chi_i, m_i')$**:** is an algorithm run by AA to update the block of a file m_i' at the index i by using sk, and it returns a new verification metadata $(\chi_i', \sigma_i', m_i')$;

$\mathcal{D}elete(sk, \chi_i, m_i)$**:** is an algorithm run by AA to delete the block m_i of a file m_i at the index i by using sk, and it returns a new verification metadata $(\chi_i', \sigma_i, \sigma_i')$; and

$\mathcal{I}nsert(sk, \chi_i, m_i)$**:** is an algorithm run by AA to insert the block of a file m_i at the index i by using sk, and it returns a new verification metadata $(\chi_i', \sigma_i', m_i')$.

To ensure the security, dynamic data operations are available only to data owners or authorized applications, who hold the secret key sk. Here, all operations are based on data blocks. Moreover, in order to implement audit services, applications need to update the index-hash tables. It is necessary for TPA and CSP to check the validity of updated data. In Fig. 5(c), we describe the process of dynamic data operations and audit. First, an authorized application obtains the public verification information ψ from TPA. Second, the application invokes the $Update$, $Delete$, and $Insert$ algorithms, and then sends the new ψ' and σ' to TPA and CSP, respectively. Next, the CSP makes use of an algorithm $Check$ to verify the validity of updated data. Note that the $Check$ algorithm is important to ensure the effectiveness of the audit. Finally, TPA modifies audit records after the confirmation message from CSP is received and the completeness of records is checked.

5 Construction for Our Scheme

We propose an efficient interactive POR (IPOR) scheme to realize the integer verification of outsourced data. This scheme includes a 3-move interactive proof protocol, which also provides privacy protection property to ensure the confidentiality of secret data.

5.1 Notations and Preliminaries

Let $\mathcal{H} = \{H_k\}$ be a keyed hash family of functions $H_k : \{0,1\}^* \rightarrow \{0,1\}^n$ indexed by $k \in \mathcal{K}$. We say that an algorithm $\mathcal{A}$ has advantage ϵ in breaking the *collision-resistance* of $\mathcal{H}$ if $\Pr[\mathcal{A}(k) = (m_0, m_1) : m_0 \neq m_1, H_k(m_0) = H_k(m_1)] \geq \epsilon$, where the probability is over random choice of $k \in \mathcal{K}$ and random bits of $\mathcal{A}$. This hash function can be obtained from hash function of BLS signatures [16].

We set up our systems using bilinear pairings proposed by Boneh and Franklin [17]. Let $\mathbb{G}$ and $\mathbb{G}_T$ be two multiplicative groups using elliptic curve conventions with a large prime order p. The function e be a computable bilinear map $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ with the following properties: for any $G, H \in \mathbb{G}$ and all $a, b \in \mathbb{Z}_p$, we have 1) Bilinearity: $e(G^a, H^b) = e(G, H)^{ab}$; 2) Non-degeneracy: $e(G, H) \neq 1$ unless G or $H = 1$; and 3) Computability: $e(G, H)$ is efficiently computable. A bilinear map system is a tuple $\mathbb{S} = \langle p, \mathbb{G}, \mathbb{G}_T, e \rangle$ composed of the objects as described above.

5.2 Proposed Construction

We present our IPOR construction in Fig. 6. In our scheme, each client holds a secret key sk, which can be used to generate the tags of many files. Each processed file produces a public verification parameter

KeyGen(1^κ): Given a bilinear map group system $\mathbb{S} = (p, \mathbb{G}, \mathbb{G}_T, e)$ and a collision-resistant hash function $H_k(\cdot)$, chooses a random $\alpha, \beta \in_R \mathbb{Z}_p$ and computes $H_1 = h^\alpha$ and $H_2 = h^\beta \in \mathbb{G}$. Thus, the secret key is $sk = (\alpha, \beta)$ and the public key is $pk = (g, h, H_1, H_2)$.

TagGen(sk, F): Splits the file F into $n \times s$ sectors $F = \{m_{i,j}\} \in \mathbb{Z}_p^{n \times s}$. Chooses s random $\tau_1, \cdots, \tau_s \in \mathbb{Z}_p$ as the secret of this file and computes $u_i = g^{\tau_i} \in \mathbb{G}$ for $i \in [1, s]$ and $\xi^{(1)} = H_\xi(\text{“}Fn\text{”})$, where $\xi = \sum_{i=1}^{s} \tau_i$ and Fn is the file name. Builds an index-hash table $\chi = \{\chi_i\}_{i=1}^{n}$ and fills out the item $\chi_i = (B_i = i, V_i = 1, R_i \in_R \{0,1\}^*)$ in χ for $i \in [1, n]$, then calculates its tag as $\sigma_i \leftarrow (\xi_i^{(2)})^\alpha \cdot g^{\sum_{j=1}^{s} \tau_j \cdot m_{i,j} \cdot \beta} \in \mathbb{G}$. where $\xi_i^{(2)} = H_{\xi^{(1)}}(\chi_i)$ and $i \in [1, n]$. Finally, sets $u = (\xi^{(1)}, u_1, \cdots, u_s)$ and outputs $\psi = (u, \chi)$ to TPA, and $\sigma = (\sigma_1, \cdots, \sigma_n)$ to CSP.

Proof(CSP, TPA): This is a 3-move protocol between Prover (CSP) and Verifier (TPA), as follows:

- **Commitment**($CSP \rightarrow TPA$): CSP chooses a random $\gamma \in \mathbb{Z}_p$ and s random $\lambda_j \in_R \mathbb{Z}_p$ for $j \in [1, s]$, and sends its commitment $C = (H_1', \pi)$ to TPA, where $H_1' = H_1^\gamma$ and $\pi \leftarrow e(\prod_{j=1}^{s} u_j^{\lambda_j}, H_2)$;
- **Challenge**($CSP \leftarrow TPA$): TPA chooses a random challenge set I of t indexes along with t random coefficients $v_i \in \mathbb{Z}_p$. Let Q be the set of challenge index coefficient pairs $\{(i, v_i)\}_{i \in I}$. TPA sends Q to CSP;
- **Response**($CSP \rightarrow TPA$): CSP calculates the response θ, μ as $\sigma' \leftarrow \prod_{(i,v_i) \in Q} \sigma_i^{\gamma \cdot v_i}$, $\mu_j \leftarrow \lambda_j + \gamma \cdot \sum_{(i,v_i) \in Q} v_i \cdot m_{i,j}$, where $\mu = \{\mu_j\}_{j \in [1,s]}$. P sends $\theta = (\sigma', \mu)$ to TPA; and
- **Check**: The verifier TPA checks whether the response is correct by

$$\pi \cdot e(\sigma', h) \stackrel{?}{=} e(\prod_{(i,v_i) \in Q} (\xi_i^{(2)})^{v_i}, H_1') \cdot e(\prod_{j=1}^{s} u_j^{\mu_j}, H_2).$$

Fig. 6. Proposed IPOR scheme for key generation, tag generation and verification protocol.

$\psi = (u, \chi)$, where $u = (\xi^{(1)}, u_1, \cdots, u_s)$, $\chi = \{\chi_i\}_{i \in [1,n]}$ is the index-hash table. We define $\chi_i = (B_i || V_i || R_i)$, where B_i is a sequence number of block, V_i is a version number of updates for this block, and R_i is a random integer to avoid collision. The value $\xi^{(1)}$ can be considered as the signature of the secret $\tau_1, \cdots, \tau_s$. Note that it must assure that ψs are different for all processed files.

In our construction, the verification protocol has 3-move structure: commitment, challenge and response. This protocol is similar to Schnorr's Σ protocol [18], which is a zero-knowledge proof system. By using this property, we ensure the verification process does not reveal anything. To prevent the leakage of stored data and tags in the verification process, the private data $\{m_{i,j}\}$ are protected by a random $\lambda_j \in \mathbb{Z}_p$ and the tags $\{\sigma_i\}$ are randomized by a $\gamma \in \mathbb{Z}_p$. Furthermore, the values $\{\lambda_j\}$ and γ are protected by the simple commitment methods, i.e., H_1^γ and $u_i^{\lambda_i} \in \mathbb{G}$, to prevent the adversaries from gaining those properties.

Moreover, it is obvious that this construction admits a short constant-size response $\theta = (\sigma', \mu) \in \mathbb{G} \times \mathbb{Z}_p^s$ without being dependent on the size of challenge. That is extremely important for large-size files.

5.3 Implementation of Dynamic Operations

To support dynamic data operations, it is necessary for TPA to employ an index-hash table χ to record the current status of the stored files. Some existing index schemes for dynamic scenarios are insecure due to replay attack on the same Hash values. To solve this problem, a simple index-hash table $\chi = \{\chi_i\}$ is used

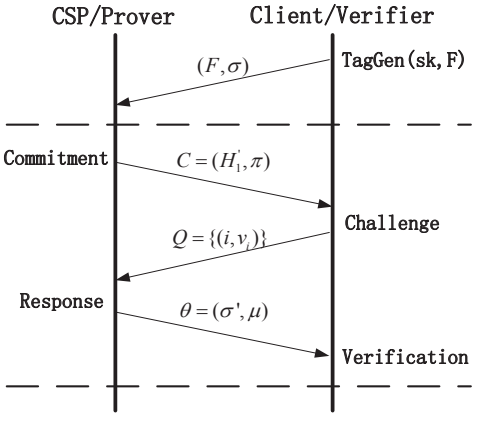


Fig. 7. Framework of IPOR model.

as described in Table 2, which includes four columns: No. denotes the real number i of data block m_i, B_i is the original number of block, V_i stores the version number of updates for this block, and R_i is a random integer to avoid collision.

In order to ensure the security, we require that each χ_i = “$B_i||V_i||R_i$” is unique in this table. Although the same values of “$B_i||V_i$” may be produced by repeating the insert and delete operations, the random R_i can avoid this collision. An alterative method is to generate an updated random value by $R_i' \leftarrow H_{R_i}(\sum_{j=1}^{s} m_{i,j}')$, where the initial value is $R_i \leftarrow H_{\xi^{(1)}}(\sum_{j=1}^{s} m_{i,j})$ and $m_i = \{m_{i,j}\}$ denotes the i-th data block. We show a simple example to describe the change of index-hash table for different operations in Table 2, where an empty record ($i = 0$) is used to support the operations on the first record. The

TABLE 2
Index-hash table with random values.

No.	B_i	V_i	R_i	
0	0	0	0	← Used to head
1	1	2	r'_1	← Update
2	2	1	r_2	
3	4	1	r_3	← Delete
4	5	1	r_5	
5	5	2	r'_5	← Insert
⋮	⋮	⋮	⋮	
n	n	1	r_n	
n+1	n+1	1	r_{n+1}	← Append

"Insert" operation on the last record is replaced with "Append" operation. It is easy to prove that each χ_i is unique in χ in our scheme.

Update(sk, ψ, m'_i)**:** modifies the version number by $V_i \leftarrow \max_{B_i=B_j}\{V_j\}+1$ and chooses a new R_i in $\chi_i \in \chi$ to get a new χ'_i; computes the new hash $\xi_i^{(2)} = H_{\xi^{(1)}}(\text{“}B_i||V_i||R_i\text{”})$; by using sk, computes $\sigma'_i = (\xi_i^{(2)})^{\alpha} \cdot (\prod_{j=1}^{s} u_j^{m'_{i,j}})^{\beta}$, where $u = \{u_j\} \in \psi$, finally outputs $O = (\chi'_i, \sigma'_i, m'_i)$.

Delete(sk, ψ, m_i)**:** computes the original σ_i by m_i and computes the new hash $\xi_i^{(2)} = H_{\xi^{(1)}}(\text{“}B_i||0||R_i\text{”})$ and $\sigma'_i = (\xi_i^{(2)})^{\alpha}$ by sk; deletes i-th record to get a new ψ'; finally outputs $O = (\chi'_i, \sigma_i, \sigma'_i)$.

Insert(sk, ψ, m'_i)**:** inserts a new record in i-th position of the index-hash table $\chi \in \psi$, and the other records move backward in order; modifies $B_i \leftarrow B_{i-1}$, $V_i \leftarrow \max_{B_i=B_j}\{V_j\} + 1$, and a random R_i in $\chi_i \in \chi$ to get a new χ'_i; computes the new hash $\xi_i^{(2)} = H_{\xi^{(1)}}(\text{“}B_i||V_i||R_i\text{”})$ and $\sigma'_i = (\xi_i^{(2)})^{\alpha} \cdot (\prod_{j=1}^{s} u_j^{m'_{i,j}})^{\beta}$, where $u = \{u_j\} \in \psi$, finally outputs $O = (\chi'_i, \sigma'_i, m'_i)$.

Check$(U/D/I, O)$**:** The application sends the operation type $U/D/I$ and the result O is given to CSP via a *secure channel*.

- For Update or Insert operations, CSP must check whether the following is held: for $(\chi'_i, \sigma'_i, m'_i)$, $e(\sigma'_i, h) \stackrel{?}{=} e(\xi_i^{(2)}, H_1) \cdot e(\prod_{j=1}^{s} u_j^{m'_{i,j}}, H_2)$; and
- For Delete operation, CSP must check whether σ_i is equal to the stored σ_i and $e(\sigma'_i, h) \stackrel{?}{=} e(H_{\xi^{(1)}}(\text{“}B_i||0||R_i\text{”}), H_1)$.

In addition, TPA must replace χ_i with the new χ'_i and check the completeness of $\chi'_i \in \psi$.

Fig. 8. Algorithms for dynamic operations.

Based on the construction of index-hash tables, we propose a simple method to provide dynamic data modification as illustrated in Fig. 8. All tags and the index-hash table should be renewed and reorganized periodically to improve the performance. Obviously, we can replace the sequent lists with dynamically linked lists to improve the efficiency of updating the index-hash table.

6 Security Analysis

First, we prove the completeness of our construction: for every available tag $\sigma \in TagGen(sk, F)$ and a random challenge $Q = (i, v_i)_{i \in I}$, the protocol always passes the verification test, that is, $\Pr[\langle CSP(F, \sigma), TPA^* \rangle(pk, \psi) = 1] = 1$. We can prove this equation holds by using Equation (1).

Next, to protect the confidentiality of checked data, we are more concerned about the leakage of private information (which includes the verified data $\{m_i\}$ and their tags $\{\sigma_i\}$) in public verification process. To address this problem, we introduce Zero-Knowledge property into our construction:

Definition 1 (Zero-knowledge): An interactive proof of retrievability scheme is computational zero knowledge if there exists a probabilistic polynomial-time algorithm S^* (call a Simulator) such that for every probabilistic polynomial-time (PPT) algorithm D and V^*, every polynomial $p(\cdot)$, and all sufficiently large s, it holds that

$$\left| \begin{array}{c} \Pr[D(pk, \psi, S^*(pk, \psi)) = 1] - \\ \Pr[D(pk, \psi, \langle P(F, \sigma), V^* \rangle(pk, \psi)) = 1] \end{array} \right| \leq 1/p(s),$$

where, $S^*(pk, \psi)$ denotes the output of simulator S. That is, for all $\sigma \in TagGen(sk, F)$, the ensembles $S^*(pk, \psi)$ and $\langle P(F, \sigma), V^* \rangle(pk, \psi)$ [3] are computationally indistinguishable.

Actually, zero-knowledge is a property that captures P's robustness against attempts to gain knowledge by interacting with it. For our IPOR scheme, we make use of the zero-knowledge property to guarantee the security of data blocks and signature tags. We have the following theorem (see Appendix A):

Theorem 1: Our IPOR scheme is a (computational) zero-knowledge provable data possession (called as ZKPOR) with respect to the polynomial-time simulators.

Then, we turn attention to the audit security for dynamic operations. It is easy to discover that the security of our scheme against dynamic operations is built on collision-resistant of all hash values $\xi_i^{(2)} = H_{\xi^{(1)}}(\chi_i)$, where $\xi^{(1)} = H_{\xi}(\text{“}Fn\text{”})$, $\xi = \sum_{i=1}^{s} \tau_i \pmod p$ and $\chi_i = \text{“}B_i||V_i||R_i\text{”} \in \chi$. Firstly, in an index-hash table $\chi = \{\chi_i\}$ and $\chi_i = \text{“}B_i||V_i||R_i\text{”}$, there exists no identical records χ_i and χ_j for all dynamic operations only if $B_i \neq B_j$, $V_i \neq V_j$, or $R_i \neq R'_j$ for any indexes $i, j \in \mathbb{N}$. Furthermore, the secrets $\{\tau_1, \cdots, \tau_s\} \in \mathbb{Z}_p^s$ are also used to avoid a collision of files that have the same file name. For both

3. The output of the interactive machine V^* after interacting with $P(F, \sigma)$ on common input (pk, ψ).

$$\begin{aligned}\pi \cdot e(\sigma', h) &= e(g,h)^{\beta \sum_{j=1}^{s} \tau_j \cdot \lambda_j} \cdot e\left(\prod_{(i,v_i)\in Q} (\xi_i^{(2)})^{v_i}, h\right)^{\alpha\cdot\gamma} \cdot e(g,h)^{\gamma\cdot\beta \sum_{j=1}^{s}(\tau_j \cdot \sum_{(i,v_i)\in Q} v_i \cdot m_{i,j})} \\ &= e(g,h)^{\beta \sum_{j=1}^{s} \tau_j \cdot \lambda_j} \cdot e\left(\prod_{(i,v_i)\in Q} (\xi_i^{(2)})^{v_i}, h\right)^{\alpha\cdot\gamma} \cdot e(g,h)^{\beta \sum_{j=1}^{s}(\tau_j \cdot \mu_j - \tau_j \cdot \lambda_j)} \\ &= e\left(\prod_{(i,v_i)\in Q} (\xi_i^{(2)})^{v_i}, h^{\alpha\cdot\gamma}\right) \cdot \prod_{j=1}^{s} e(u_j^{\mu_j}, h^{\beta}). \end{aligned} \quad (1)$$

mechanisms, we can prove the following theorem (see Appendix B):

Theorem 2: The hash values $\xi_i^{(2)}$ is $(\varepsilon, \sqrt{2^{L+2}p}\ln\frac{1}{1-\varepsilon})$ collision-resistant in our scheme [4] , even if a client generates $\sqrt{2p \cdot \ln\frac{1}{1-\varepsilon}}$ files with the same file name, and the client repeats $\sqrt{2^{L+1} \cdot \ln\frac{1}{1-\varepsilon}}$ times to modify, insert and delete data blocks, where the collision probability is at least ε, $\tau_i \in \mathbb{Z}_p$, and $|R_i| = L$.

Based on collision-resistant of χ, we consider a new notion of security for our scheme on dynamic operations, which is called dynamic existential unforgeability under an adaptive chosen message attack [19], [20]. This kind of security can be defined using the following game between a challenger $\mathcal{B}$ and an adversary $\mathcal{A}$:

1) Initial: Given a file F, the challenger $\mathcal{B}$ simulates $KeyGen(1^\kappa)$ and $TagGen(sk, F)$ to generate the public parameters pk and ψ, and sends them to the adversary $\mathcal{A}$;

2) Learning: $\mathcal{A}$ adaptively issues at most q_t times queries $q_1, \cdots, q_t$ to learn the information of tags via dynamic operations, as follows:

- **a) Update query** (i, m_i')**:** $\mathcal{B}$ generates $(\chi_i', \sigma_i', m_i') \leftarrow Update(sk, \chi_i, m_i')$ and sends it to $\mathcal{A}$;
- **b) Delete query** (i)**:** $\mathcal{B}$ generates $(\chi_i', \sigma_i, \sigma_i') \leftarrow Detele(sk, \chi_i, m_i)$ and sends it to $\mathcal{A}$;
- **c) Insert query** (i, m_i')**:** $\mathcal{B}$ generates $(\chi_i', \sigma_i', m_i') \leftarrow Insert(sk, \chi_i, m_i)$ and sends it to $\mathcal{A}$;

At any time, $\mathcal{A}$ can query the hash values $\xi_i^{(2)} = H_{\xi^{(1)}}(\chi_i)$ on at most q_h records of its choice $\{\chi_i\}$ and $\mathcal{B}$ responds with random values.

3) Output: Eventually, $\mathcal{A}$ outputs a forgery $(\chi_i^*, m_i^*, \sigma_i^*)$ and wins the game if:

- $(\chi_i^*, m_i^*, \sigma_i^*)$ is not any of q_t queries; and
- $(\chi_i^*, m_i^*, \sigma_i^*)$ is a valid tag for $\xi_i^{(2)}$, that is, $e(\sigma_i^*, h) = e(\xi_i^{(2)}, H_1) \cdot e(\prod_{j=1}^{s} u_j^{m_{i,j}^*}, H_2)$.

An adversary $\mathcal{A}$ is said to (ε, q_t, q_h)-break our scheme if $\mathcal{A}$ makes at most q_t tag queries for dynamic operations and q_h hash queries, as well as success probability of game with at least ε. A scheme is (ε, q_t, q_h)-dynamically and existentially unforgeable under an adaptively chosen message attack if there exists no forgery that is susceptible to (ε, q_t, q_h)-break.

We note that the above-mentioned defintion captures a stronger version of existential unforgeability than the standard one on signature schemes, as it requires that the adversary cannot even generate a new tag on a previously signed message. Based on this game, we prove that our scheme is (ε, q_t, q_h)-dynamically and existentially unforgeable for the Computational Diffie-Hellman (CDH) problem (see Appendix C), as follows:

Theorem 3: Given the (ε', q')-CDH in a cyclic group $\mathbb{G} \in \mathbb{S}$ with an order p, our audit scheme is an (ε, q_t, q_h)-dynamically existentially unforgeable to resist the attacks of tag forgery for dynamic data operations with random oracle model, whenever $q_t + q_h \leq q'$ and all ε satisfy $\varepsilon \geq \varepsilon'/(1 - \frac{q_t}{p})$ if each χ_i is collision-resistant in index-hash table χ.

7 Performance and Evaluation

It is obvious that enormous audit activities would increase the computation and communication overheads of our audit service. However, the less frequent activities may not detect anomalies in a timely manner. Hence, the scheduling of audit activities is significant for improving the quality of audit services. In order to detect anomalies in a low-overhead and timely manner, we attempt to optimize the audit performance from two aspects: performance evaluation of probabilistic queries and scheduling of periodic verification. Our basic idea is to maintain a tradeoff between overhead and accuracy, which helps us improve the performance of audit systems.

7.1 Probabilistic Queries Evaluation

The audit service achieves the detection of CSP servers' misbehaviors in a random sampling mode to reduce the workload on the server. The detection probability P of disrupted blocks is an important parameter to guarantee that these blocks can be detected in a timely manner. Assume TPA modifies e blocks out of the n-block file. The probability of disrupted blocks is $\rho_b = \frac{e}{n}$. Let t be the number of queried blocks for

4. We use the terminology (ε, Q) to denote an algorithm with average-case success probability ε, in which the number of oracle queries made by the algorithm is at most Q.

a challenge in the protocol proof. We have detection probability $P = 1 - (\frac{n-e}{n})^t = 1 - (1-\rho_b)^t$. Hence, the number of queried blocks is $t = \frac{\log(1-P)}{\log(1-\rho_b)} \approx \frac{P \cdot n}{e}$ for a sufficiently large n.[5] This means that the number of queried blocks t is directly proportional to the total number of file blocks n for the constant P and e. In Fig. 9, we show the results of the number of queried blocks under different detection probabilities (from 0.5 to 0.99), different number of file blocks n (from 10 to 10,000), and constant number of disrupted blocks (10 for $n <$ 1,000 and 100 for $n \geq$ 1,000).

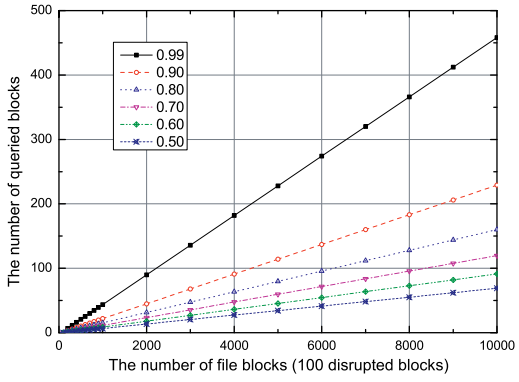


Fig. 9. Number of queried blocks under different detection probabilities and different numbers of file blocks.

We observe the ratio of queried blocks in the total file blocks $w = \frac{t}{n}$ under different detection probabilities. Based on our analysis, it is easy to determine that this ratio holds since $w = \frac{t}{n} = \frac{\log(1-P)}{n \cdot \log(1-\rho_b)} \approx \frac{P}{e}$. However, the estimation of w is not an accurate measurement. To clearly represent this ratio, Fig. 10 plots w for different values of n, e and P. It is obvious that the ratio of queried blocks tends to be a constant value for a sufficiently large n. For instance, in Fig. 10(a) if there exist 100 disrupted blocks, the TPA asks for $w = 4.5\%$ and 2.3% of n ($n > 1,000$) in order to achieve P of at least 99% and 90%, respectively. However, this ratio w is also inversely proportional to the number of disrupted blocks e. For example, in Fig. 10(b) if there exist 10 disrupted blocks, the TPA needs to ask for $w = 45\%$ and 23% of n ($n > 1,000$) in order to achieve the same P, respectively. It demonstrates our audit scheme is effective under the higher probability of disrupted blocks.

Note that, instead of probability verification, our IPOR scheme also supports absolute integrity verification, in which TPA checks all file blocks in a challenge query $Q = \{(i, v_i)\}_{i \in I}$, that is, $I = [1, n]$. Furthermore, in order to shorten the length of challenge query Q, we simply send a random $seed$ to CSP, and then CSP generates the challenge index coefficient pair $(i, v_i) = (i, H_{seed}(i))$ for all $i = [1, n]$, where $H_k(\cdot)$ is a collision-resistant hash function.

5. In terms of $(1-\frac{e}{n})^t = 1 - \frac{e \cdot t}{n}$, we have $P = 1 - (1 - \frac{e \cdot t}{n}) = \frac{e \cdot t}{n}$.

7.2 Schedule of Periodic Verification

Too frequent audits may waste the network bandwidth and computing resources of TPA and CSPs. However, less frequent audits would not be conducive to detect the exceptions in a timely manner. Thus, it is necessary to disperse the audit tasks (in which the total number of queried blocks is evaluated as we discussed in the previous section) throughout the entire audit cycle so as to balance the overload and increase the difficulty of attacks in a relatively short period of time.

The sampling-based audit has the potential to significantly reduce the workload on the servers and increase the audit efficiency. Firstly, we assume that each audited file has an audit period T, which depends on how important it is for the owner. For example, a common audit period may be assigned as one week or one month, and the audit period for important files may be set as one day. Of course, these audit activities should be carried out at night or on weekend.

Also, we make use of the audit frequency f to denote the number of occurrences of an audit event. This means that the number of TPA's queries is $T \cdot f$ in an audit period T. Our evaluation indicates the detection probability $P = 1 - (1 - \rho_b)^{n \cdot w}$ in each audit event. Let P_T denotes the detection probability in an audit period T. Hence, we have the equation $P_T = 1 - (1 - P)^{T \cdot f}$. Given $1 - P = (1 - \rho_b)^{n \cdot w}$, the detection probability P_T can be denoted as $P_T = 1 - (1-\rho_b)^{n \cdot w \cdot T \cdot f}$. Based on this equation, TPA can obtain the probability ρ_b depending on the transcendental knowledge of the cloud storage provider. Moreover, the audit period T can be predefined by a data owner in advance. Hence, the above equation can be used to analyze the parameter values w and f. It is obvious to obtain the equation $f = \frac{\log(1-P_T)}{w \cdot n \cdot T \cdot \log(1-\rho_b)}$.

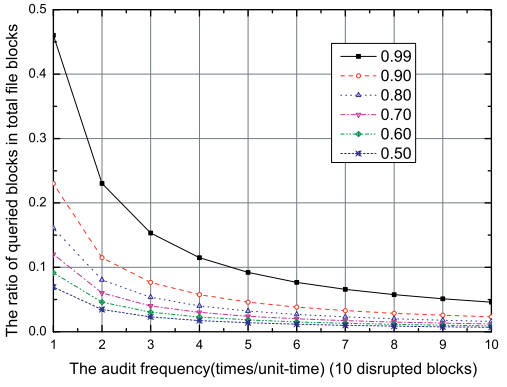


Fig. 11. Ratio of queried blocks in total file blocks under different audit frequency for 10 disrupted blocks and 10,000 file blocks.

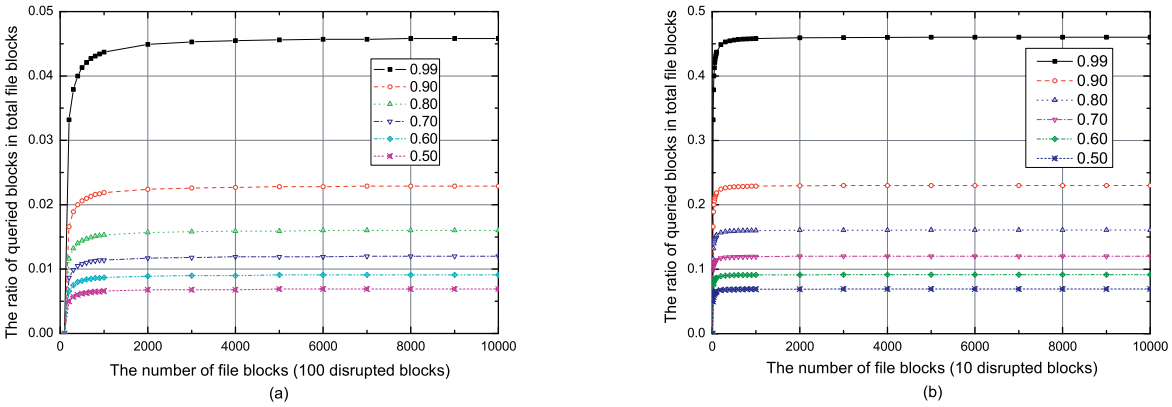


Fig. 10. Ratio of queried blocks under different detection probabilities and different number of disrupted blocks.

This means that the audit frequency f is inversely proportional to the ratio of queried blocks w. That is, with the increase of verification frequency, the number of queried blocks decreases at each verification process. In Fig. 11, we show the relationship between f and w under 10 disrupted blocks for 10,000 file blocks. We can observe a marked drop of w along with the increase of frequency.

In fact, the relationship between f and w is comparatively stable for P_T, ρ_b, and n due to $f \cdot w = \frac{\log(1-P_T)}{n \cdot T \cdot \log(1-\rho_b)}$. TPA should choose an appropriate frequency to balance the overhead. For example, if $e = 10$ blocks in 10,000 blocks ($\rho_b = 0.1\%$), then TPA asks for 658 blocks and 460 blocks for $f = 7$ and 10 to achieve at least 99% of P_T. Hence, an appropriate audit frequency would greatly reduce the sampling numbers, as well as computation and communication overheads of an audit service.

7.3 Implementation and Experimental Results

To validate our approaches, we have implemented a prototype public audit service. Our prototype utilizes three existing services/applications: Amazon Simple Storage Service (S3), which is an untrusted data storage server; a local application server, which provides our audit service; and an existing open source project called Pairing-Based Cryptography (PBC) library upon which to build our prototype. We present some details about these three components as follows:

Storage service: Amazon Simple Storage Service (S3) is a scalable, pay-per-use online storage service. Clients can store a virtually unlimited amount of data, paying for only the storage space and bandwidth that they are using, without an initial start-up fee. The basic data unit in S3 is an object, and the basic container for objects in S3 is called a bucket. In our example, objects contain both data and meta-data (tags). A single object has a size limit of 5 GB, but there is no limit on the number of objects per bucket. Moreover, a script on Amazon Elastic Compute Cloud (EC2) is used to provide the support for verification protocol and dynamic data operations.

Audit service: We used a local IBM server with two Intel Core 2 processors at 2.16 GHz running Windows Server 2003. Our scheme was deployed in this server, and then the server performs the integrity check in S3 storage, conforming the assigned schedule via 250 MB/sec of network bandwidth. A socket port was also opened to support the applications' accesses and queries for the audit service.

Prototype software: Using GMP and PBC libraries, we have implemented a cryptographic library upon which temporal attribute systems can be constructed. These C libraries contain approximately 5,200 lines of codes and were tested on both Windows and Linux platforms. The elliptic curve utilized in our experiments is a MNT curve, with a base field size 159 bits and the embedding degree 6. The security level is chosen to be 80 bit, which means $|p| = 160$.

Firstly, we quantified the performance of our audit scheme under different parameters, such as file size sz, sampling ratio w, and sector number per block s. Our analysis shows that the value of s should grow with the increase of sz to reduce computation and communication costs. Thus, experiments were carried out as follows: the stored files were chosen from 10KB to 10MB, the sector numbers were changed from 20 to 250 in terms of the file size, and the sampling ratios were also changed from 10% to 50%. The experimental results are shown in Fig. 12(a). These results indicate that computation and communication costs grow slowly with increase of file size and sampling ratio.

Next, we compared the performance of each activity in our verification protocol. It is easy to derive theoretically that the overheads of "commitment" and "challenge" resemble one another, and the overheads of "response" and "verification" also resemble one another. To validate such theoretical results, we changed

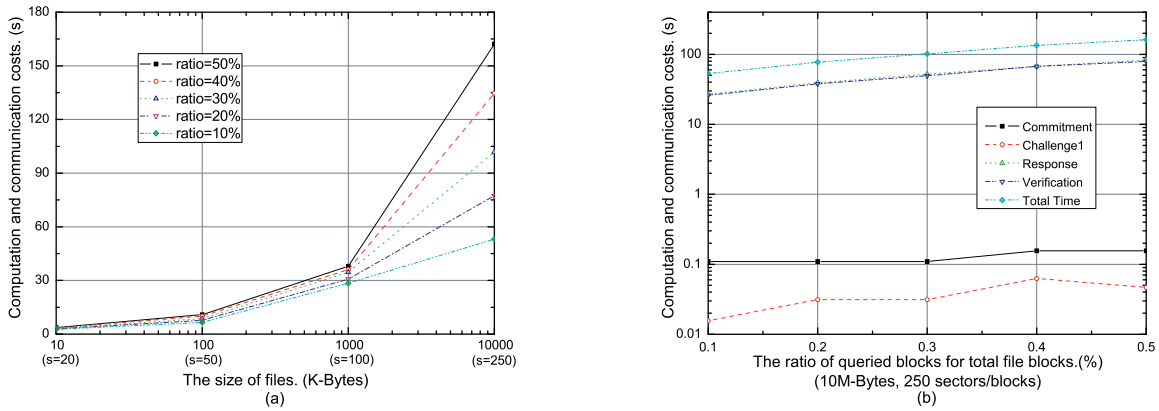


Fig. 12. Experiment results under different file size, sampling ratio, and sector number.

the sampling ratio w from 10% to 50% for a 10MB file and 250 sectors per block. In Fig. 12(b), we show the experiment results, in which the computation and communication costs of "commitment" and "challenge" are slightly changed for sampling ratio, but those for "response" and "verification" grow with the increase of sampling ratio.

Then, in the Amazon S3 service, we set that the size of block is 4K bytes and the value of s is 200. Our experiments also show that, in TagGen phase, the time overhead is directly proportional to the number of blocks. Ideally, this process is only executed when a file is uploaded into the S3 service. The verification protocol can be run in approximately constant time. Similarly, three dynamic data operations can be performed in approximately constant time for any block.

Finally, reducing the communication overheads and average workloads is extremely critical for an efficient audit schedule. With probabilistic algorithm, our scheme is able to realize the uniform distribution of verified sampling blocks based on the security requirements of clients, as well as the dependabilities of storage services and running environments. In our experiments, we make use of a simple schedule to periodically manage all audit tasks. The results show that audit services based on our scheme can support a great deal of audit tasks, and the performance of scheduled audits are more preferable than the straightforward individual audit.

8 Conclusions

In this paper, we presented a construction of dynamic audit services for untrusted and outsourced storages. We also presented an efficient method for periodic sampling audit to enhance the performance of third party auditors and storage service providers. Our experiments showed that our solution has a small, constant amount of overhead, which minimizes computation and communication costs.

Acknowledgments

The work of Yan Zhu and Shimin Chen was partially supported by the National Development and Reform Commission under Project "A cloud-based service for monitoring security threats in mobile Internet". This work of Gail-J. Ahn and Hongxin Hu was partially supported by the grants from US National Science Foundation (NSF-IIS-0900970 and NSF-CNS-0831360) and Department of Energy (DE-SC0004308). This work of Stephen S. Yau and Ho G. An was partially supported by the grants from US National Science Foundation (NSF-CCF-0725340).

References

[1] Amazon.com, "Amazon s3 availability event: July 20, 2008," Online at http://status.aws.amazon.com/s3-20080720.html, July 2008.

[2] A. Juels and B. S. K. Jr., "Pors: proofs of retrievability for large files," in *Proceedings of the 2007 ACM Conference on Computer and Communications Security, CCS 2007*, 2007, pp. 584–597.

[3] M. Mowbray, "The fog over the grimpen mire: Cloud computing and the law," HP Lab., Tech. Rep. HPL-2009-99, 2009.

[4] A. A. Yavuz and P. Ning, "Baf: An efficient publicly verifiable secure audit logging scheme for distributed systems," in *ACSAC*, 2009, pp. 219–228.

[5] G. Ateniese, R. C. Burns, R. Curtmola, J. Herring, L. Kissner, Z. N. J. Peterson, and D. X. Song, "Provable data possession at untrusted stores," in *Proceedings of the 14th ACM Conference on Computer and Communications Security*, 2007, pp. 598–609.

[6] G. Ateniese, R. D. Pietro, L. V. Mancini, and G. Tsudik, "Scalable and efficient provable data possession," in *Proceedings of the 4th international conference on Security and privacy in communication netowrks, SecureComm*, 2008, pp. 1–10.

[7] C. C. Erway, A. Küpçü, C. Papamanthou, and R. Tamassia, "Dynamic provable data possession," in *Proceedings of the 16th ACM Conference on Computer and Communications Security*, 2009, pp. 213–222.

[8] H. Shacham and B. Waters, "Compact proofs of retrievability," in *Advances in Cryptology - ASIACRYPT 2008*, ser. Lecture Notes in Computer Science, J. Pieprzyk, Ed., vol. 5350. Springer, 2008, pp. 90–107.

[9] H.-C. Hsiao, Y.-H. Lin, A. Studer, C. Studer, K.-H. Wang, H. Kikuchi, A. Perrig, H.-M. Sun, and B.-Y. Yang, "A study of user-friendly hash comparison schemes," in *ACSAC*, 2009, pp. 105–114.

[10] A. R. Yumerefendi and J. S. Chase, "Strong accountability for network storage," in *FAST*. USENIX, 2007, pp. 77–92.

Enabling Data Integrity Protection in Regenerating-Coding-Based Cloud Storage: Theory and Implementation (Supplementary File)

Henry C. H. Chen and Patrick P. C. Lee

1 Additional Related Work

This section supplements Section 2 of the main paper with other related studies.

Data integrity protection is first considered in memory management systems [8]. In online memory checking, a user checks whether each read/write operation in an unreliable memory device is correct, using a small, trusted (and possibly private) memory device. Naor and Rothblum [16] improve the efficiency of online memory checking. Instead of reading all bits of a file during checking, they sample bits from the file. The sampling idea is also used in our data checking scheme.

Proof of retrievability (POR) [15] and *proof of data possession* (PDP) [3] are recent works that explore efficient data integrity checking (i.e., through sampling instead of whole-file checking) in single-server archival storage systems. POR [15] embeds a set of pseudorandom blocks into an encrypted file stored in the server, and the client can later check if the server keeps the pseudorandom blocks. Error correcting codes are also included in the stored file to allow recovery of a small amount of errors within a file. However, the number of checks that the client can issue is limited by the number of the embedded random blocks. On the other hand, PDP [3] allows the client to keep a small amount of metadata. The client can then challenge the server against a set of random file blocks to see if the server returns the proofs that match the metadata on the client side. Both schemes can further minimize the network transfer bandwidth by aggregating proofs into smaller messages. However, such aggregation techniques require the servers to have certain encoding capabilities.

Several follow-up studies on POR and PDP improve their computation and communication complexities (e.g., [10], [12], [22]). Adding protection of *dynamic* files (i.e., files that can be updated after being stored) to PDP is considered in [4], [13]. Some studies focus on the public verifiability of efficient integrity checking schemes (e.g., [5], [20], [23]).

Multi-server (or multi-cloud) storage has been proposed and implemented to protect against data loss [6], [9], [14], [19] and mitigate vendor lock-ins [1]. Oggier et al. [17] consider exact regenerating codes with collaborative repairs, and verify the integrity of fully-regenerated chunks with trusted hashes to guard against adversarial corruptions. Zhu et al. [26] propose a cooperative PDP scheme for multi-cloud storage, but it is difficult to deploy the scheme in practice as it requires the cooperation of different multiple cloud providers.

2 Illustrations of FMSR Codes

This section supplements Section 3.1 of the main paper with additional illustrations of FMSR codes on NCCloud [14]. Figure 1(a) illustrates the entire file upload process in NCCloud using FMSR codes for $n = 4$ and $k = 2$, while Figure 1(b) shows how a file is encoded inside NCCloud using matrix multiplication.

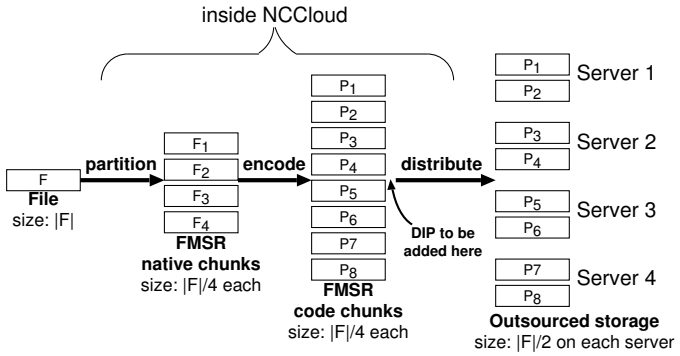
3 Illustrations of FMSR-DIP Codes

This section supplements Section 4 of the main paper with additional illustrations of FMSR-DIP codes. First, we summarize the notations used in Table 1. Figure 2 shows an overview of how we augment an FMSR code chunk into an FMSR-DIP code chunk.

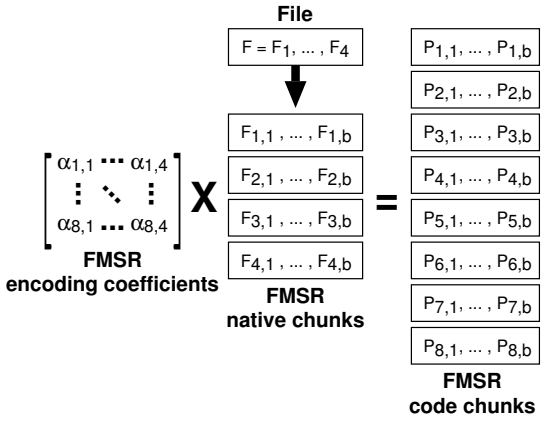
4 Aggregating Rows in Thick Cloud Storage

This section supplements Section 4.1 of the main paper. Our work assumes the thin-cloud setting in which servers only need to support basic read/write operations. Here, we briefly remark on how FMSR-DIP codes can be modified to utilize server-side encoding capabilities when such functionalities are available in thick

• *H. Chen and P. Lee are with the Department of Computer Science and Engineering, The Chinese University of Hong Kong (Emails: {chchen,pclee}@cse.cuhk.edu.hk)*



(a) File upload process



(b) Encoding process using matrix multiplication

Fig. 1. An example of the file upload and encoding operations in NCCloud using a (4,2)-FMSR code. Data is stored in four servers, in which the data of any two servers suffice to recover the original file. Each server stores two code chunks of total size $|F|/2$.

TABLE 1
A summary of the key notations.

Symbols	Meaning
n, k	Parameters for FMSR codes
n', k'	Parameters for AECC
b	FMSR chunk size
b'	FMSR-DIP chunk size
F	File to be backed up
$\|F\|$	Original file size
$\{F_i\}$	FMSR native chunks (i.e., partitions of original file)
$\{P_i\}$	FMSR code chunks
$\{P'_i\}$	FMSR-DIP code chunks
$\{\alpha_{ij}\}$	FMSR encoding coefficients
λ	Checking percentage

cloud storage services. Our goal is to aggregate the downloaded bytes during the Check operation, so as to reduce the amount of data transfer.

The idea is as follows. During the Check operation, instead of downloading all the randomly selected bytes from the servers, we can divide the random indices into groups and request each server to return the XOR-sum of all the selected bytes on a group-by-group basis for each chunk. Thus, for each group, we download one byte from each code chunk. Due to the *distributive* nature

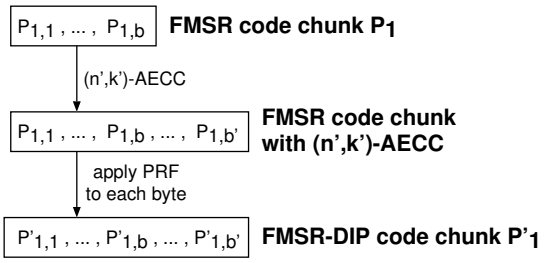


Fig. 2. An overview of how an FMSR code chunk P_1 is augmented into an FMSR-DIP code chunk P'_1.

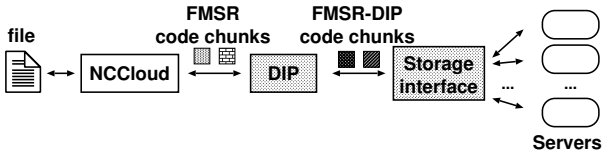


Fig. 3. Integration of the DIP scheme into NCCloud.

of finite field operations, *the XOR-sum of the selected bytes of the code chunks can be decoded to the XOR-sum of the corresponding bytes of the native chunks*. We can then perform rank checking on the XOR'ed bytes instead.

To illustrate, suppose that in the Check operation, we apply the XOR-sum to the r_1th and r_2th rows of the FMSR-DIP code chunks, i.e., $\{P'_{ir_1} \oplus P'_{ir_2}\}_{1 \le i \le n(n-k)}$. First, we remove the PRF, i.e., $P_{ir_1} \oplus P_{ir_2} = P'_{ir_1} \oplus P'_{ir_2} \oplus \mathsf{PRF}(i||r_1) \oplus \mathsf{PRF}(i||r_2)$. Note that:

$$\begin{aligned} P_{ir_1} \oplus P_{ir_2} &= \textstyle\sum_{j=1}^{k(n-k)} \alpha_{ij} F_{jr_1} \oplus \sum_{j=1}^{k(n-k)} \alpha_{ij} F_{jr_2} \\ &= \textstyle\sum_{j=1}^{k(n-k)} \alpha_{ij} (F_{jr_1} \oplus F_{jr_2}). \end{aligned}$$

If $P_{ir_1} \oplus P_{ir_2}$ is error-free, then it can be correctly decoded into the XOR-sum of the r_1th and r_2th rows of bytes of the native chunks. We then apply rank checking to verify $P_{ir_1} \oplus P_{ir_2}$ as before.

We can further reduce data transfer by allowing servers to XOR bytes across different chunks. Note that the XOR-sum of the r_1th and r_2th row of different chunks P_i and $P_{i'}$ (that is $P_{ir_1} \oplus P_{ir_2} \oplus P_{i'r_1} \oplus P_{i'r_2}$) is the following:

$$\begin{aligned} &\left(\textstyle\sum_{j=1}^{k(n-k)} \alpha_{ij} (F_{jr_1} \oplus F_{jr_2})\right) \oplus \left(\textstyle\sum_{j=1}^{k(n-k)} \alpha_{i'j} (F_{jr_1} \oplus F_{jr_2})\right) \\ &= \textstyle\sum_{j=1}^{k(n-k)} (\alpha_{ij} \oplus \alpha_{i'j})(F_{jr_1} \oplus F_{jr_2}). \end{aligned}$$

5 Additional Implementation Details

This section supplements Section 5 of the main paper with additional implementation details.

5.1 Integration of DIP into NCCloud

We implement a standalone DIP module and a storage interface module, and integrate them with NCCloud as shown in Figure 3. In the Upload operation, NCCloud generates code chunks for a file based on FMSR codes. The code chunks will be temporarily stored in the local filesystem instead of being uploaded to the servers as

in [14]. The DIP module then reads the FMSR code chunks from the local filesystem, encodes them with DIP, and passes the resulting FMSR-DIP code chunks to the storage interface module, which will upload the FMSR-DIP code chunks to multiple servers (or a cloud-of-clouds [1], [6], [14]). In the Download operation, the DIP module checks the integrity of the chunks retrieved from the servers before relaying the chunks to NCCloud for decoding. Note that we can issue a range GET request to download a selected range of bytes.

Our DIP module mostly operates on a per-chunk basis. Thus, it can harness parallelism in today's multi-core technologies by concurrently encoding different code chunks. For example, a (4,2)-FMSR code will create eight FMSR code chunks, each of which can be encoded into an FMSR-DIP code chunk with a dedicated process. In an eight-core machine, we can have up to eight-fold speed-up over the sequential approach. This can significantly speed up the entire DIP operation.

Our current implementation uses a *modular* approach that separates the FMSR code module (i.e., NCCloud) and the DIP module. It is possible to combine the two modules into a single design to reduce the overhead, so as to eliminate the passing of FMSR code chunks between NCCloud and our DIP module using a local staging directory. In addition, we may exploit certain inherent properties of such combination to further reduce the computational overhead, and we pose this issue as future work. On the other hand, our modular approach allows us to *flexibly* enable DIP on demand in real deployment.

5.2 Instantiating Cryptographic Primitives

We implement all cryptographic operations using OpenSSL 1.0.0g [18]. All cryptographic primitives use 128-bit secret keys. The primitives are instantiated as described below.

Symmetric encryption. We use AES-128 in cipher-block chaining (CBC) mode.

Pseudorandom function (PRF). We use AES-128 for PRF. The PRF input is first transformed to a plaintext block, which is then encrypted with AES-128.

Pseudorandom permutation (PRP). Our PRP implementation is based on AES-128, but applied in a different way as in PRF. Note that the domain size of the PRP is the number of elements to be permuted. To implement a PRP with a small and flexible domain size, we follow the approach in Method 1 of [7]. We first create a list of indices from 0 to $d-1$, where d is the desired domain size of our PRP. Then we encrypt each index in turn with AES-128 and sort the encrypted indices. Finally, the permutation is given by the positions of the original indices in the sorted list of encrypted indices. A more efficient way can be used to generate a small PRP [10], at the expense of a larger storage overhead.

Message authentication codes (MACs). We use HMAC-SHA-1 to compute MACs.

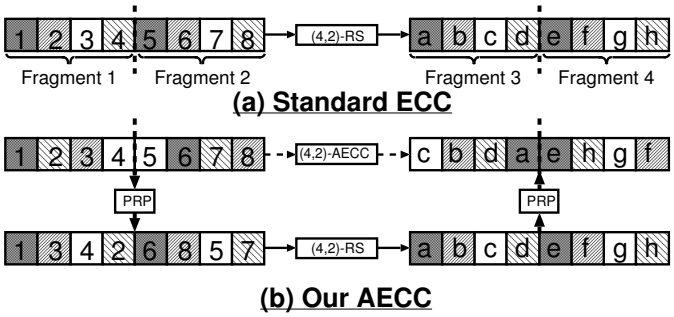


Fig. 4. Comparing standard ECC and our AECC (using for example (4,2)-Reed-Solomon encoding). The bytes of the same shade correspond to the same stripe.

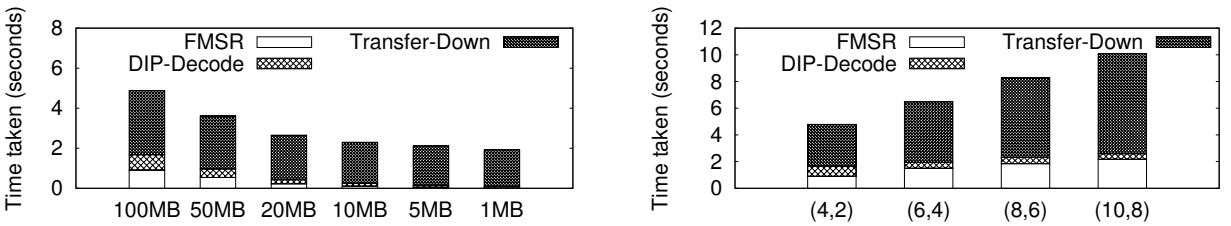
Adversarial error-correcting codes (AECCs). We apply the systematic AECC adapted from [10], [11] with two main differences. First, for efficiency, we do not encrypt the AECC parities, since we will apply PRF to the entire DIP-encoded chunk after applying AECC. PRF itself serves as an encryption. Second, and most notably, instead of applying a single PRP to the entire code chunk, we first divide the code chunk into k' fragments, and apply a different PRP to each fragment. The secret key of the PRP for each fragment is formed by the XOR-sum of a master PRP secret key and the fragment number. Applying PRP to a fragment rather than a chunk reduces the domain size and hence the overall memory usage. The trade-off is that we reduce the security protection. Also, our approach is more resilient to burst errors since each byte of a stripe is confined to its own fragment, while in permuting over the entire chunk, a stripe may have many of its bytes clustered together.

Figure 4 shows our AECC implementation, in which we use zfec [25] for the underlying systematic ECC (based on Reed-Solomon codes). We first apply a PRP to each of the k' fragments within the FMSR code chunk. We then apply systematic ECC to the permuted chunk (divided into b/k' stripes of k' bytes each), where the ith stripe ($1 \leq i \leq b/k'$) comprises the bytes in the ith positions of all fragments. Finally, we permute each fragment of the ECC parities, and append the permuted parities to the code chunk.

6 Uses of Security Primitives

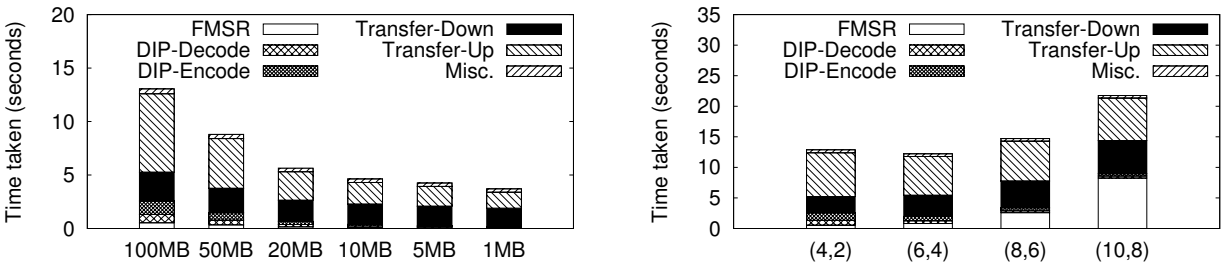
This section supplements Section 6 of the main paper with additional details about the effects of the various security primitives used in FMSR-DIP codes.

Pseudorandom function (PRF). The effect of applying PRF on the data is similar to encrypting the data. It randomizes the data so that it is infeasible for the adversary to manipulate the original data and hence corrupt the data in such a way that the corrupted bytes form consistent systems of linear equations during the Check operation. PRF is important for guarding against a *mobile*



(a) Different file sizes (b) Different (n, k) values of FMSR codes

Fig. 5. Running time of the entire Download operation on a local cloud.



(a) Different file sizes (b) Different (n, k) values of FMSR codes

Fig. 6. Running time of the entire Repair operation on a local cloud.

Byzantine adversary [9], which can possibly corrupt data on all servers over time via creeping corruption [9].

To illustrate the necessity of PRFs in our construction, consider the following attack on our construction in the absence of PRFs. Remember that our adversary is mobile, and thus can zero out bytes at a specific offset on all chunks (i.e., a row) across multiple epochs. Note that a random linear combination of zeroes is always zero, so a row of zeroes is always consistent and decodable (to give zeroes). If the adversary corrupts only a few rows at a time, it would be near impossible for us to detect the attack before too many rows are corrupted, rendering our data irrecoverable.

Symmetric encryption. We encrypt the metadata to hide the encoding coefficients of FMSR codes. This protects against the scenario where the PRF values can be recovered with known encoding coefficients and original file content.

Adversarial error-correcting codes (AECC). We use AECC to randomize the stripe structure, so that it is infeasible for the adversary to deterministically render chunks unrecoverable.

Message authentication codes (MAC). We include the MACs of individual chunks as metadata, and replicate them to all servers to allow integrity verification of any chunks.

7 Additional Evaluation Results

This section supplements Section 7 of the main paper. Here, we present evaluation results of the running time overhead of FMSR-DIP codes in the Download and Repair operations. We further analyze the monetary cost overhead with the pricing models of different commercial cloud providers.

7.1 Running Time Overhead of Download and Repair

Figures 5 and 6 show the running times of Download and Repair, respectively. Under the (4,2)-FMSR code, in Download, the DIP-Decode part accounts for 3.80% (for 1MB) to 15.7% (for 100MB) of the overall Download time, while in Repair, the DIP-Decode and DIP-Encode parts altogether account for 3.19% (for 1MB) to 15.7% (for 100MB) of the overall Repair time. Using a 100MB file and varying (n, k) for FMSR codes, in Download, the DIP-Decode part accounts for 4.14% (for (10,8)-FMSR) to 15.8% (for (4,2)-FMSR) of the overall Download time, while in Repair, the DIP-Decode and DIP-Encode parts altogether account for 3.24% (for (10,8)-FMSR) to 15.8% (for (4,2)-FMSR) of the overall Repair time.

Execution Anomaly Detection in Distributed Systems through Unstructured Log Analysis

Qiang FU[1], Jian-Guang LOU[1], Yi WANG[2], Jiang LI[1]

[1]Microsoft Research Asia
Beijing, P.R. China
{qifu, jlou, jiangli}@microsoft.com

[2]Dept. of Computer Science & Technology
Beijing University of Posts and Telecommunications
Beijing, P.R. China
wangyi.tseg@gmail.com

Abstract -- **Detection of execution anomalies is very important for the maintenance, development, and performance refinement of large scale distributed systems. Execution anomalies include both work flow errors and low performance problems. People often use system logs produced by distributed systems for troubleshooting and problem diagnosis. However, manually inspecting system logs to detect anomalies is unfeasible due to the increasing scale and complexity of distributed systems. Therefore, there is a great demand for automatic anomaly detection techniques based on log analysis. In this paper, we propose an unstructured log analysis technique for anomaly detection. In the technique, we propose a novel algorithm to convert free form text messages in log files to log keys without heavily relying on application specific knowledge. The log keys correspond to the log-print statements in the source code which can provide cues of system execution behavior. After converting log messages to log keys, we learn a Finite State Automaton (FSA) from training log sequences to present the normal work flow for each system component. At the same time, a performance measurement model is learned to characterize the normal execution performance based on the log messages' timing information. With these learned models, we can automatically detect anomalies in newly input log files. Experiments on Hadoop and SILK show that the technique can effectively detect running anomalies.**

Keywords - log analysis; distributed system; problem diagnosis; FSA

I. INTRODUCTION

Large scale distributed systems are becoming key engines of IT industry. For a large commercial system, execution anomalies, including erroneous behavior or unexpected long response times, often result in user dissatisfaction and loss of revenue. These anomalies may be caused by hardware problems, network communication congestion or software bugs in distributed system components. Most systems generate and collect logs for troubleshooting, and developers and administrators often detect anomalies by manually checking system printed logs. However, as many large scale and complex applications are deployed, manually detecting anomalies becomes very difficult and inefficient. At first, it is very time consuming to diagnose through manually examining a great amount of log messages produced by a large scale distributed system. Secondly, a single developer or system administrator may not have enough knowledge of the whole system, because many large enterprise systems often make use of Commercial-Off-the-Shelf components (e.g. third party components). In addition, the increasing complexity of distributed systems also lowers the efficiency of manual problem diagnosis further. Therefore, developing automatic execution anomaly monitoring and detection tools becomes an essential requirement of many distributed systems to ensure the Quality of Service.

There are two classes of typical anomalies: one is work flow errors - errors occurring during the execution paths; the other is execution low performance - the execution time takes much longer than normal cases although its execution path is correct. In this paper, we present an unstructured log analysis technique that can automatically detect system anomalies using commonly available system logs. It requires neither additional system source code instrumentation nor any runtime code profiling. The technique mainly consists of two processes: the learning process and the detection process. The goal of the learning process is to obtain models that represent the normal execution behavior of the system from those logs produced by normally completed jobs. The input data for the learning process is training log files printed by different machines. At first, we convert the log message sequences in the log files into log key sequences. Log keys are obtained by abstracting log messages. Then, we a derive Finite State Automaton (FSA) to model the execution path of the system. With the learned FSAs, we can identify the corresponding state sequences from training log sequences. Next, we count the execution time of each state transition in state sequences, and obtain a performance measurement model through statistical analysis. In the detection process, for newly input log sequences, we check them with those learned models to automatically detect anomalies. It should be noticed that the system's normal behavior may change after an upgrade. Therefore, it is necessary to re-train the model after each system upgrade.

Assumptions: In our technique, system anomaly detection is based on the cues gained from the previous normally completed jobs' log files. We assume that

each log message has a corresponding time stamp that indicates its generation time. We further assume that the logs are recoded using thread IDs or request IDs to distinguish logs of different threads or work flows. Most modern operating systems (such as Windows and Linux) and platforms (such as Java and .NET) provide thread IDs. We can therefore work with sequential logs only.

The paper is organized as follows. In section 2, several related research efforts are briefly surveyed. The log key extraction and FSA construction are introduced in section 3 and section 4. In section 5, we discuss the performance measurement model construction. After that, anomaly detection is described in section 6. Then, experimental results are presented in section 7. Finally, section 8 concludes the paper.

II. RELATED WORK

Monitoring and maintaining techniques that make use of execution logs are the least invasive and most applicable, because execution logs are often available during a system's daily running. Therefore, analyzing logs for problem diagnosis has been an active research area for several decades. In this paper, we only survey the approaches that perform the analysis automatically.

One set of algorithms [1, 2, 3, 4] judge the job's trace sequence as a whole, where a log sequence is often simply recognized as a symbol string. Dickenson et al [1] collect execution profiles from program runs, and use classification techniques to categorize the collected profiles based on some string distance metrics. Then, an analyst examines the profiles of each class to determine whether or not the class represents an anomaly. Mirgorodskiy et al [2] also use string distance metrics to categorize function-level traces, and identify outlier traces or anomalies that substantially differ from the others. Yuan et al [4] propose a supervised classification algorithm to classify system call traces based on the similarity to the traces of known problems. In other literature, a quantitative feature is extracted from each log sequence for error detection. For example, in [3], the authors preprocess the logs to extract the number of log occurrence times as a log feature, and detect anomalies using principal component analysis (PCA). These kinds of algorithms can find whether the job is abnormal, while can hardly obtain the insight and accurate information about abnormal jobs.

Another set of algorithms [5-8] view system logs as a series of footprints of systems' execution. They try to learn FSA models from the traces to model the system behavior. In the work of Cotroneo et al [5], FSA models are first derived from the traces of Java Virtual Machine collected by the JVMMon tool [6]. Then, logs of unsuccessful workloads are compared with the inferred FSA models to detect anomalous log sequences. SALSA [7] examines Hadoop logs to construct FSA models of the Datanode module and TaskTracker module. In [8], based on the traces that record the sequences of components traversed in a system in response to a user request, the authors construct varied-length n-grams and a FSA to characterize the normal system behavior. A new trace is compared against the learned FSA to detect whether it is abnormal. In their algorithm, a varied-length n-gram represents a state of the FSA. Unlike these methods, which heavily depend on application specific knowledge including some predefined log tokens and the stage structure of Map-Reduce, our algorithm can work in a black-box style. In addition, our algorithm is the only one that uses timing information in the log sequence to detect the low performance problem.

In some other literature [17, 18], logs are used to perform troubleshooting related tasks in different scenarios. GMS [17] detects abnormal machines with wrong configurations. It extracts features from the data source and applies the distributed HilOut algorithm to identify the outliers as the misconfigured machines. Its data source includes log files, utility statistics and configuration files. In [18], a decision tree is learned to identify the causes of detected failures where the failures have been detected beforehand. It records the runtime properties of each request in a multi-tier Web server, and applies statistical learning techniques to identify the causes of failures. Unlike them, our algorithm mainly tries to detect anomalies through exploiting the timing and circulation information.

III. LOG KEY EXTRACTION

Systems logs usually record run-time program behaviors, including events, states and inter-component interactions. An unstructured log message often contains two types of information: one type is free-form text string that is used to describe the semantic meaning of a recorded program behavior; the other type is a parameter that is used to express some important system attributes. In general, the number of different log message types is often huge or even infinite because of various parameter values. Therefore, during log data mining, directly considering log messages as a whole may lead to the curse of dimension.

In order to overcome this problem, we replace each log message by its corresponding log key to perform analysis. The log key is defined as the common content of all log messages which are printed by the same log-print statement in the source code. In other words, a log key equals to the free-form text string of the log-print statement without any parameters. For example, the log key of log message 5 (shown in Figure 1) is "***Image file of size saved in seconds***". We analyze logs based on log keys because: (1) In general cases, different log-print statements often output different log text messages. It means that each type of log key corresponds to one specific log-print statement in the source code. Therefore, a sequence of log keys can reveal the execution path of the program. (2) The number of log key types is

finite and is much less than the number of log message types. It can help us to avoid the curse of dimension during data mining.

The challenging problem is that we know neither which log messages are printed by the same log-print statement nor where parameters are in log messages. Therefore, it is very difficult to identify log keys. Generally, the log messages printed by the same statement are often highly similar to each other, while two log messages printed by different log-print statements are often quite different. Based on this observation, we can use clustering techniques to group log messages printed by the same statement together, and then find their common part as the log key.

However, the parameters may cause some clustering mistakes because the log messages printed by different statements may also be similar enough if they contain a lot of identical parameter values. In order to reduce the parameters' influence on clustering, we first erase the contents that are obvious parameter values according to some empirical knowledge. Then, we further apply a raw log key clustering and group splitting algorithm to obtain log keys. Figure 1 gives an example to illustrate the procedure of extracting log keys from log messages.

A. Erasing parameters by empirical rules

As we know, parameters are often in forms of numbers, URIs, IP addresses; or they follow the special symbols such as the colon or equal-sign; or they are embraced by braces, square brackets, or Parentheses. These contents can be easily identified. Therefore, empirical rules are often used to recognize and remove these parameters [9]. By roughly going through the log files, we can define some empirical regular expression rules to describe those typical parameter cases, and erase the matched contents. After that, the left contents of log messages are defined as raw log keys. The second block of Figure 1 gives some examples of raw log keys. We can see that the IP addresses, the numbers, and the full path of a file are all removed from the log messages.

Although many parameters are erased, there are still some parameters that could not be completely removed in raw log keys. The main reason is that the empirical rules can't exhaust all parameter patterns without application specific knowledge.

B. Raw log key clustering

We separate a raw log key into words using a space as separator. We use words as primitives to represent raw log keys because words are minimal meaningful elements in a sentence. So, each raw log key can be represented as a word sequence.

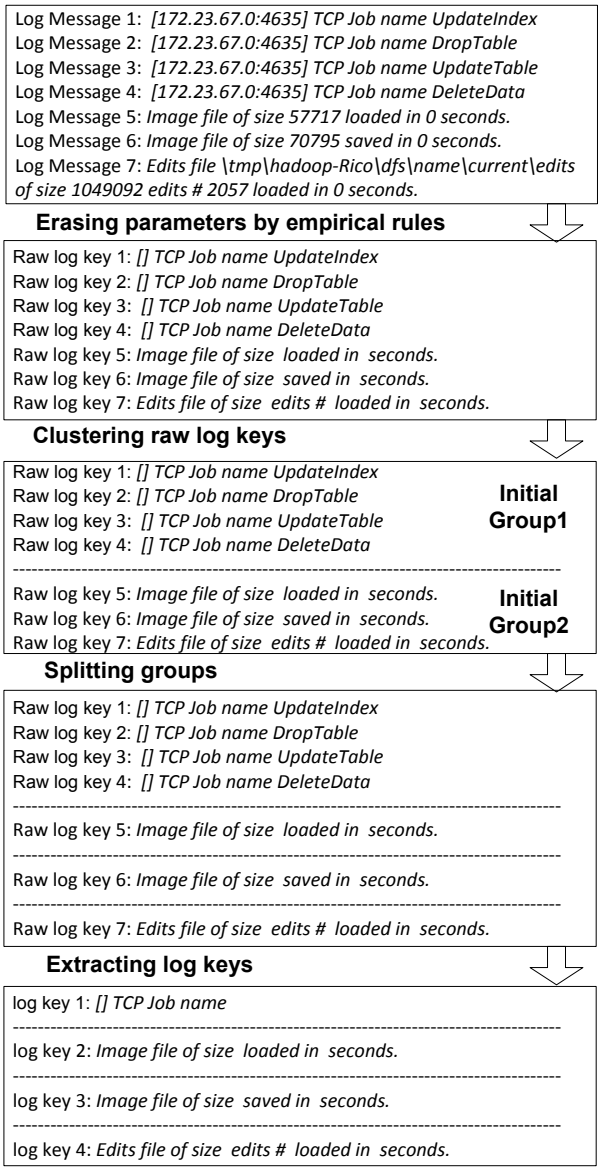
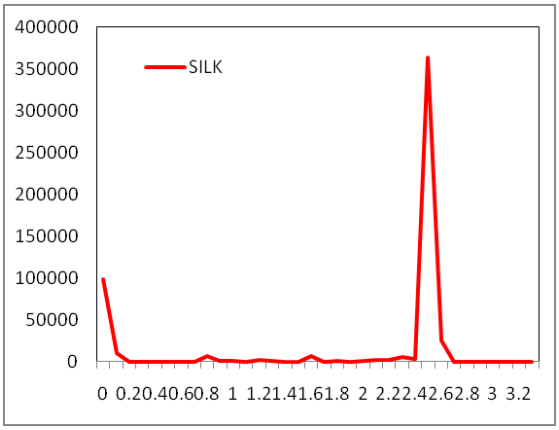


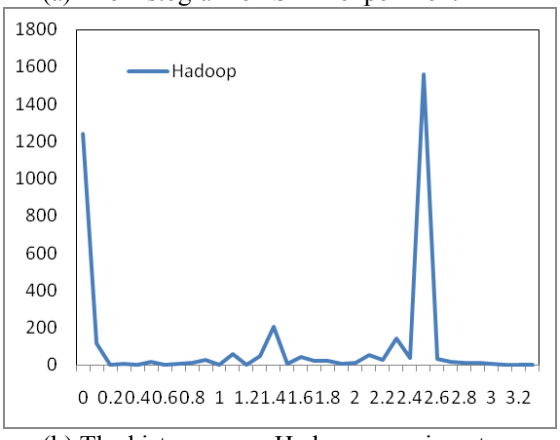
Figure 1 Examples of log key extraction

Before clustering, we need to find a proper metric to represent the similarity of two raw log keys. The string edit distance is a widely used metric to represent the similarity between word sequences. It equals to the number of edit operations required to transform one word sequence to the other. One edit operation can operate only one word. The operation can be adding, deleting or replacing. Obviously, the edit distance only counts the number of operated words; it does not consider the positions of the operated words. However, for our problem, the positions of operated words in raw log keys are meaningful for measuring similarity. It is because most programmers tend to write text messages (log keys) firstly, and then add parameters afterwards. Therefore, words at the beginning of raw log keys have

more probability to be parts of log keys than words at the end of raw log keys do. Therefore, the operated word at the beginning of the raw log keys should be more significant for measuring raw log keys' difference. Based on this observation, we measure raw log keys' similarity by the weighted edit distance, in which we use sigmoid similar function to compute weights at different positions. For two raw log keys rk_1 and rk_2, we denote the necessary operations required to transform rk_1 to rk_2 as $OA_1, OA_2, \ldots, OA_{EO}$; EO is the number of necessary operations. The weighted edit distance between rk_1 and rk_2 is denoted as $WED(rk_1, rk_2)$, $WED(rk_1, rk_2) = \sum_{i=1}^{EO} \frac{1}{1+e^{(x_i - \upsilon)}}$. Here, x_i is the index of the word that is operated by the i^{th} operation OA_i; υ is a parameter controlling weight function.



(a) The histogram on SILK experiment



(b) The histogram on Hadoop experiment

Figure 2. The histogram of raw log key pair number over weighted edit distance

We cluster similar raw log keys together. For every two log keys, if the weighted edit distance between them is smaller than a threshold ς, we connect them with a link. Then, each connected component corresponds to a group which is called as an initial group. The initial group examples are shown in the third block in Figure 1.

The threshold ς could be automatically determined according to the following procedure. For every two raw log keys, we compute the weighted edit distance between them. Then we obtain a set of distance values. Each distance should be either inner-class distance or inter-class distance. The inner-class (or inter-class) distance is the distance between two raw log keys corresponding to the same log key (or different log keys). In general, the inner-class distances are usually small while the inter-class distances are large. Therefore, we use a k-means clustering algorithm to cluster all distances into two groups. The distances in the two groups roughly correspond to the inner-class and the inter-class distances respectively. Finally, we select the largest distance from the inner-class distance group as the value of threshold ς.

We obtain the raw log keys by the experiments on Hadoop and SILK respectively (the experiments' details are described in section 7). We calculate the distances between every two raw log keys, and show the histogram of raw log key pair number over distance in Figure 2. The x-coordinate is the value of the weighted edit distance. The y-coordinate is the number of raw log key pairs. The figures show that: (1) There are two significant peaks in each histogram. It seems that the proposed weighted edit distance is a good similarity metric for raw log key clustering. (2) There is a flat region between two peaks. It implies that our raw log key clustering algorithm is not sensitive to the threshold ς.

C. *Group splitting*

Ideally, raw log keys in the same initial group correspond to the same log key. In such cases, a log key can be obtained by extracting the common part of the raw log keys in the same initial group. However, raw log keys in one initial group may correspond to different log keys because those log keys are similar enough. To handle those cases, we propose a group splitting algorithm to obtain log keys.

For an initial group, suppose there are GN raw log keys in this group. The common word sequence of the raw log keys within the group could be represented by $CW_1, CW_1, \ldots, CW_N$. For example, the initial group 2 in Figure 1 contains raw log key 5, 6, 7, and the common word sequence in the raw log keys are "***file***", "***of***", "***size***", "***in***", "***seconds***".

For each of the raw log keys in this group, e.g. the i^{th} log key, the common word sequence $CW_1, CW_2, \ldots, CW_N$ separates the raw log key into N+1 parts which is denoted as $DW_1^i, DW_2^i, \ldots, DW_N^i, DW_{N+1}^i$, where DW_j^i $(2 \le j \le N-1)$ is the i^{th} raw log key's content between CW_{j-1} and CW_j; DW_1^i is the i^{th} raw log key's content on the left side of CW_1; DW_{N+1}^i is the i^{th} raw log key's content on the right side of CW_N. We call DW_j^i as the private content at position j of the i^{th} raw log key. In the above example, the private content sequence of raw log key 7 is "***Edits***", $\emptyset$, $\emptyset$, "***edits*** #

loaded", ∅,∅. In the paper, ∅ represents that there is not any word in the private content.

For each position j, $1 \le j \le N+1$, we can obtain GN private contents at position j from GN raw log keys in the group, and they are DW_j^1, DW_j^2, …, DW_j^{GN}. We denote the number of different values (not including ∅) among those GN values as VN_j, and VN_j is called the private number at position j. For the initial group 2 in Figure 1, VN_1 =2, VN_2 =0, VN_3 =0, VN_4 =3, VN_5 =0, VN_6=0.

Intuitively speaking, if the private contents at position j are parameters, VN_j is often a large number because parameters may probably have many different values. However, if the private contents at position j are a part of log keys, VN_j should be a small number. Based on this observation, we find the smallest positive one among VN_1, VN_2,…, VN_N, VN_{N+1}, e.g. VN_J. If VN_J is equal to or bigger than a threshold ϱ, which means that the private contents at position J have at least ϱ different values, then we consider that the private contents at position J are parameters. In such a situation, this initial group does not split anymore. Otherwise, if VN_J is smaller than the threshold ϱ, we consider that the private contents at position J are a part of log keys. In such a situation, this initial group splits into VN_J sub-groups, satisfying that the raw log keys in the same sub-group have the same private content at position J. In the paper, we set ϱ as 4 according to experiments.

For the initial group 2, VN_1 is the smallest positive value 2 and is smaller than the threshold 4, so the initial group 2 splits into 2 sub-groups according to raw log keys' private contents at position 1. The raw log key 5 and 6 are in one sub-group, because they have the same private content "***Image***"; the raw log key 7 is in the other sub-group.

When there are multiple private numbers at different positions that have the same smallest positive value smaller than the threshold, we further compare the entropies at those positions respectively, select the one position with the minimal entropy, and split the group according to the private contents at that position. We denote the entropy at position j as EP_j. We compute EP_j according to the distribution of private content values at position j. For example, for the initial group 2 and j=1, we can obtain 3 values of the private content which are "***Image***", "***Image***", and "***Edits***". The value's distribution is p("***Image***")=2/3, p("***Edits***")=1/3, so $EP_1 = -\frac{2}{3}\log\frac{2}{3} - \frac{1}{3}\log\frac{1}{3} = 0.918$. The entropy rule is reasonable because a smaller entropy indicates lesser diversity, which means the private contents at that position have more possibility to be parts of log keys.

If there are still multiple positions that have the same private number and the same entropy, then we split the group according to the private contents at the most left one among those positions.

We perform the split procedure repeatedly, until there is no group satisfying the split condition. Finally, we extract the common part of raw log keys in each group as a log key.

D. Determine log keys for new log messages

After the above steps, we obtain the log key set from the training log messages in the training log files. When a new log message comes, we determine its log key according to the following two steps: First, we use the empirical rules to extract the raw log key from the log message. Second, we select the log key which has the minimal edit distance to the raw log key of the log message. If the weighted edit distance between the raw log key and the selected log key is smaller than a threshold σ, the selected log key is considered as the log key of the log message. Otherwise, the log message is considered as an error log message, and its log key is its raw log key. Here, we set σ as the largest one of the weighted edit distances between all raw log keys of training log messages and their corresponding log keys.

By replacing each log message with its corresponding log key, a log message sequence can be converted into a log key sequence.

IV. WORK FLOW MODEL

In order to detect anomalies of work flows, we use a Finite State Automaton (FSA) to model the execution behavior of each system module. Although there are some other alternate models, such as Petri-Net, we adopt FSA because it is simple but effective. FSA has been widely used in testing and debugging software applications [11]. A FSA consists of a finite number of states and transitions between the states. A set of algorithms have been proposed in previous literature to learn FSA from sequential log sequences [10, 11, 12]. In this paper, we use the algorithm proposed by [11] to learn a FSA for each system component from training log key sequences which are produced by normally completed jobs. Each transition in the learned FSAs corresponds to a log key. All training log key sequences can be interpreted by the learned FSAs. Therefore, each training log key sequence can be mapped to a state sequence. Figure 3 shows the example of the learned FSM of JobTracker of Hadoop (refer to Section 7.1). We give the state interpretations according to the log message in Table 1. From the learned the FSM, we obtain the following work flow: from S87 to S96, the JobTracker carries out some initialization tasks when a new job is submitted. After initialization, the state machine enters S197 to add a new Map/Reduce task. For each map task, it selects local or remote data source for processing. Then, the task is completed. When the last task is finished, the job is completed, and all resources of tasks are cleared iteratively. In fact, the learned FSM correctly reflects the real work flow of the JobTracker.

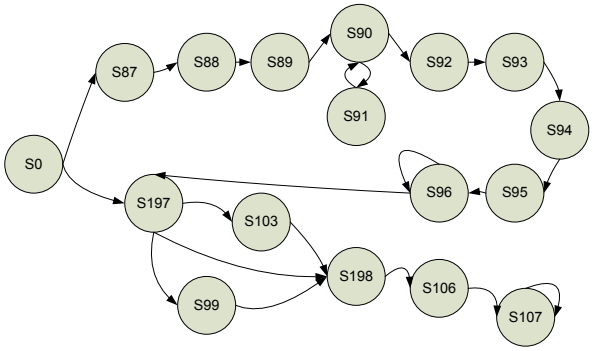


Figure 3. Example of a learned FSM

Table 1. The interpretations of states

State	Interpretation
S87~ S96	Initialization when a new job submitted
S197	Add a new map/reduce task
S103	Select remote data source
S99	Select local data source
S198	Task complete
S106	Job complete
S107	Clear task resource

V. PERFORMANCE MEASUREMENT MODEL

In this section, we present our technique to characterize the performance of the normally completed jobs. By comparing with normal performance characteristics, we can detect low performance in new jobs.

After log key extraction, we obtain corresponding log key sequences. The time stamp of a log key is the same as the time stamp of its corresponding log message. In order to derive a performance measurement model, we need to know applications' execution states. Therefore, we first convert each log key sequence to its corresponding state sequence. A state's time stamp is specified by the time stamp of its corresponding log key in the log key sequence.

In a system execution, there are two types of low performance problems. One is that the time interval that a system component transits from a state to the next state is much longer than normal cases; we name it transition time low performance. The other is that the circulation numbers of a loop structure are far more than normal cases; we name that loop low performance. We use the transition time between adjacent states and the circulation numbers of all loop structures to characterize the normal performance of jobs.

A. Transition time measurement model

In a distributed system, each machine writes log message sequences to its local disc independently. Therefore, different training state sequences may be derived from logs in different machines. Suppose we have *M* machines in a distributed system. For each state transition in the FSA, e.g. from S_a to S_b, the time intervals between two adjacent states (S_a, S_b) in the training state sequences produced by i^{th} machine are denoted as $\tau_i^1(S_a,S_b)$, $\tau_i^2(S_a,S_b)$,…, $\tau_i^{K_i}(S_a,S_b)$; $1 \le i \le M$. Here, K_i is the total number of the time intervals in all state sequences produced by the i^{th} machine.

We use a Gaussian model to present the distribution of the state transition interval. In practice, the computational capacity of machines in a distributed system is often heterogeneous. The different computing capacity of machines results in the state transition time intervals in different machines being quite different. In order to handle this problem, we introduce a capacity parameter for each machine. Our model contains machine independent Gaussian distribution parameters $\{\mu(S_a,S_b),\ \sigma^2(S_a,S_b)\}$ and machine dependent capacity parameters $\{\lambda_1(S_a,S_b),\lambda_2(S_a,S_b),\ldots,\ \lambda_M(S_a,S_b)\}$. Here, the Gaussian distribution $N(\mu(S_a,S_b),\sigma^2(S_a,S_b))$ is used to represent the distribution of the state transition time on an imaginary computer with a standard computing capacity. It is only determined by the property of the specified state transition, and does not depend on the property of any specific machine. The computers' properties are modeled by the computers' computing capacity parameters $\lambda_i(S_a,S_b)$, $1 \le i \le M$. The computers' computing capacity parameters are also associated with the state transition, because different state transitions often correspond to different computing tasks and the same computer may have a different computing capacity under different work load characteristics.

In this subsection, because the state transition is specified, we abridge state indicators in expressions or formulas for simplicity. We assume that the mean value of state transition time in the i^{th} machine is proportional to its computing capacity parameter λ_i, and the variance is proportional to λ_i^2. With that assumption, the obtained transition time instances in the i^{th} machine satisfy the Gaussian distribution $N(\lambda_i\mu,(\lambda_i\sigma)^2)$, $1 \le i \le M$. We further assume that the obtained transition time instances are independent, and then the likelihood function is as follows.

$$p\left(\tau_1^1,\tau_1^2,\ldots,\tau_1^{K_1},\tau_2^1,\tau_2^2,\ldots,\tau_2^{K_2},\ldots,\tau_M^1,\ldots,\tau_M^{K_M}\right) = \prod_{i=1}^{M}\left[\prod_{j=1}^{K_i} N(\tau_i^j;\lambda_i\mu,(\lambda_i\sigma)^2)\right] \quad (1)$$

With the variable substitutions of $\alpha_i = \lambda_i\mu$ and $\beta = \frac{\sigma^2}{\mu^2}$, we can obtain the log-likelihood function:

$$L(\alpha_1,\alpha_2,\ldots,\alpha_M,\beta) = -\sum_{i=1}^{M}\sum_{j=1}^{K_i}[2\ln\alpha_i + \ln\beta + \frac{1}{\beta}(1-\frac{\tau_i^j}{\alpha_i})^2] \quad (2)$$

According to the Maximum Likelihood Estimation criterion, the optimal parameters should maximize $L(\alpha_1,\alpha_2,\ldots,\alpha_M,\beta)$. Because the optimal parameters should satisfy that the partial differentiates of $L(\alpha_1,\alpha_2,\ldots,\alpha_M,\beta)$ equal to 0, we have:

$$\begin{cases} \alpha_i = \dfrac{\sqrt{\left(\sum_{j=1}^{K_i}\tau_i^j\right)^2 + 4K_i\beta\left(\sum_{j=1}^{K_i}{\tau_i^j}^2\right)} - \left(\sum_{j=1}^{K_i}\tau_i^j\right)}{2K_i\beta}, 1 \le i \le M \\ \beta = \left(\sum_{i=1}^{M}\sum_{j=1}^{K_i}(\alpha_i - \tau_i^j)^2\right) / \left(\sum_{i=1}^{M} K_i {\alpha_i}^2\right) \end{cases} \quad (3)$$

However, there is no closed form solution to the above equation group; we can only use an iterative procedure to obtain an approximation of the optimal parameters. It could be proved that after each iteration step, the value of $L(\alpha_1, \alpha_2, \ldots, \alpha_M, \beta)$ increases. The iterative procedure is shown in Table 2. When the difference of β in two iterations is small enough ($< \mathrm{Th}_\beta$), the iterative procedure terminates.

Finally, we can obtain the transition time measurement model: the transition time from S_a to S_b in the i^{th} machine satisfies the Gaussian distribution $N(\alpha_i(S_a, S_b), \alpha_i^2(S_a, S_b)\beta(S_a, S_b))$.

It should be pointed out that the above algorithm can be easily implemented in a parallel mode. According to formula (3), when given β, α_i can be determined by the sample data in the i^{th} machine, i.e. τ_i^j $(1 \le j \le K_i)$. Thus, each α_i can be calculated separately at the i^{th} machine. When given α_i $(1 \le i \le M)$, the intermediate results, i.e. ${K_i}{\alpha_i}^2$ and $\sum_{j=1}^{K_i}(\alpha_i - \tau_i^j)^2$, can also be calculated by machines separately. Then, it is very easy to integrate those intermediate results to obtain β. Therefore, our algorithm can be used to learn models from the logs of very large scale systems.

B. Circulation numbers measurement model

The circulation numbers of loop structures are meaningful measurements for low performance detection because some executions' low performance is caused by abnormally more loops although each of its adjacent state transition times seem normal. A loop structure is defined as a directed cyclic chain composed by the state transition in the learned FSA. For example, for the FSA shown in Figure 4, one loop structure is $\{S_2, S_3\}$, the other is $\{S_1, S_2, S_3\}$. A loop structure execution instance is formed by consecutively repeating several rounds of a loop structure from its beginning to its end; and the number of execution rounds is defined as a circulation number. For example, in the state sequence "$S_0 S_1 S_2 S_3 S_2 S_3 S_1 S_2 S_3 S_4$", the subsequences, e.g. "$S_2 S_3 S_2 S_3$" and "$S_2 S_3$", are two execution instances of the loop structure $\{S_2, S_3\}$ in the state sequence, and the circulation numbers are 2 and 1 respectively; the subsequence, e.g. "$S_1 S_2 S_3 S_2 S_3 S_1 S_2 S_3$" is an execution instance of the loop structure $\{S_1, S_2, S_3\}$, and the circulation number is 2.

We identify loop structures in the learned FSA. For each loop structure, e.g. L, we find the execution instances of L in all training state sequences, and record their circulation numbers as $C^1(L), C^2(L), \ldots, C^{H(L)}(L)$; where $H(L)$ is the amount of L's execution instances. Similarly, we use Gaussian distribution $N(\mu(L), \sigma^2(L))$ to model them.

$$\mu(L) = \frac{1}{H(L)}\sum_{i=1}^{H(L)} C^i(L) \quad (4)$$

$$\sigma^2(L) = \frac{1}{H(L)}\sum_{i=1}^{H(L)} [C^i(L) - \mu(L)]^2 \quad (5)$$

Table 2. Iterative procedure to compute parameters

Initialization:

$\alpha_i = \frac{1}{K_i}\sum_{j=1}^{K_i}\tau_i^j$, $1 \le i \le M$;

$\beta = \beta' = 0$;

While *true*

Set $\beta' = \beta$;

Using current value of α_i $(1 \le i \le M)$, compute β according to the last one formula in the formula group (3);

If $|\beta' - \beta| < Th_\beta$,

break;

Else

using current value of β, compute α_i $(1 \le i \le M)$ according to the first M formula in the formula group (3);

Endif

End

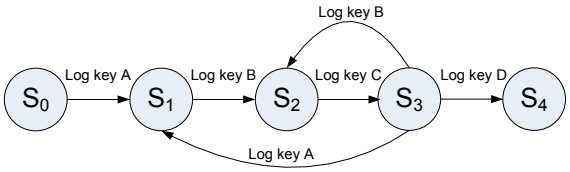


Figure 4. An example of FSA

VI. ANOMALIES DETECTION

For a newly input log message sequence, we can obtain the corresponding log key sequence according to section 3.4. If the log key sequence can be generated by the learned FSA, then we consider that there is no work flow error. Otherwise, the first log key in the sequence that can't be generated by the learned FSA is detected as a work flow error. The details of work flow error detection can be found in paper [11]. In this paper, we mainly focus on the low performance detection.

A. Transition time low performance detection

During low performance detection, we first convert the testing log key sequences to the corresponding state sequences according to the learned FSA. For each state transition in the state sequence produced by the i^{th} machine, e.g. from S_a to S_b, we then compare its execution time with the learned transition time measurement model of the i^{th} machine. If the execution time is larger

than a threshold $\gamma_i(S_a, S_b)$, it is considered as a transition time low performance. Here, the threshold is defined as the sum of the mean value and ϵ times standard deviation of the learned transition time distribution.

$$\gamma_i(S_a, S_b) = \alpha_i(S_a, S_b)(1 + \epsilon\sqrt{\beta(S_a, S_b)}) \quad (6)$$

Obviously, the smaller ϵ is, the more state transitions are detected as low performance problems. At the same time, there will be more false positives and less false negatives. When applying our technique, users can adjust the value of ϵ according to real requirements. In the experiments, we set ϵ as 3.

B. Loop low performance detection

Similar to transition low performance detection, for each loop structure L, we calculate its threshold $\vartheta(L)$ as follows

$$\vartheta(L) = \mu(L) + \epsilon\sigma(L) \quad (7)$$

We find the execution instances of L whose circulation numbers are larger than $\vartheta(L)$ as loop low performance.

VII. EXPERIMENTS

In this section, we evaluate the proposed technique through detecting anomalies in two typical distributed computing systems: Hadoop and SILK (a privately owned distributed computing system). In this section, we represent some typical cases to demonstrate our technique, and give out some over all evaluations on our experiment results.

A. Case study on Hadoop

Hadoop [13] is a well-known open-source implementation of Google's Map-Reduce [14] framework and distributed file system (GFS)[15]. It enables distributed computing of large scale, data-intensive and stage-based parallel applications. Hadoop is designed with master-slave architecture. NameNode is a master of the distributed file system, which manages the metadata of all stored data chunks, while DataNodes are slaves used to store the data chunks. JobTracker acts as a task scheduler that decomposes the job into smaller tasks and assigns the tasks to different TaskTrackers. A TaskTracker is a worker of a task instance.

The logs produced by Hadoop are not sequential log message sequences in its original forms. The log messages for different tasks interleave together. However, we can easily extract sequential log message sequences from logs by the task IDs.

Table 3. Basic configurations of machines

Machine	Basic configuration
PT03~PT05	Intel dual-core E3110@3.0G, 8G RAM
PT06~PT11	Intel quad-core E5430@2.66G, 8G RAM
PT12~PT17	AMD Quad-Core 2376@2.29G, 8G RAM

Our test bed of Hadoop (version 0.19) contains 16 machines (from PT3 to PT17) connected with a 1G Ethernet switch. The basic configurations are listed in Table 3. Among them, PT17 is used as a master that hosts NameNode and JobTracker components. The others are used as slaves, and each slave hosts DataNode and TaskTracker components. During the experiments, we run the stand-alone program (namely CPUEater) which consumes a predefined ratio of CPU so that we can better simulate a heterogeneous environment. Table 4 shows the utility ratios (i.e. 100%-consumed CPU ratio of CPUEater) and the learned model parameters. We can see that the more powerful machine, the smaller the average transition time is.

Table 4. Utility ratio and model parameters

Machine	Utility Ratio	Learned parameters	
		α (s)	β
pt09	100%	38.04	0.0187
pt07	30%	47.10	
pt12	50%	65.02	
pt14	30%	65.63	
pt05	50%	78.46	

In the learning stage, we run 100 jobs of counting words in the test bed and collect the produced log files of these jobs as training data. The counting words job gives out the word frequency in the input text files. Each input text file for a job is about 10G. In the testing stage, we run 30 counting words jobs to produce testing data.

In this subsection, we give one example of the test cases in Table 5. In this case, we manually insert a low performance problem by limiting the bandwidth of machine PT9 to 1Mbps when running a job, and check whether our algorithm can detect it. The result shows that our algorithm can successfully detect the low performance problem that the transition time from state #21 to state #1 is much larger than the normal cases (i.e. 60s > 38.04s).

Table 5. Low performance transition of Hadoop

Time Stamp	State ID	State Meaning
2009-01-18 10:42:31.452	21	Data source for a Map task is selected.
2009-01-18 10:43:30.423	1	Map task is completed.

B. Case study on SILK

SILK is a distributed system developed by our lab for large scale data intensive computing. Unlike MapReduce, SILK uses a Directed Acyclic Graph (DAG) framework similar to Dryad [16]. SILK is also designed based on the master-slave architecture. A Scheduler-Server component works as a master to decompose the job into smaller tasks, and then schedule and manage the tasks. SILK produces many log files during execu-

tion. For example, it generates about 1 million log messages every minute (depending on workload intensity) in a 256-machine system. Each log message contains a process ID and a thread ID. We can group log messages with the same process ID and thread ID into sequential log sequences. The test bed of SILK contains 7 machines (1 master, 6 slaves), which is set up for daily-build testing. As our training data, we collect the training log files of all successful jobs during a ten-day running in the test-bed. The test logs are generated during one month of daily-build testing. Our algorithm can detect several system execution anomalies (shown in Table 7). In this subsection, we give two typical examples.

Case 1: In this case, due to a networking issue, a slave task (CopyDatabase) tries several times to connect to a database, which makes a response to the master (SchedulerServer) and is largely delayed. From the log sequence of the master, our algorithm finds that the transition time from state #424 to state #428 is much larger than expected (see Table 6). According to the learned model, the average time interval is 12.32s, while the time interval in this case is 42.53s. Therefore, our algorithm detects it as an anomaly of transition time low performance.

Table 6. Case 1: Low performance transition of SILK

Time Stamp	State ID	State Meaning
2008-09-09 18:44:52.749	424	Job task is started.
2008-09-09 18:45:35.280	428	A worker progress event is received.

Case 2: In this case, the master (SchedulerServer) sends a job finish message to a client, but the client never replies. This causes the master to repeat the attempt more than 20 times before giving up. Compared with 1 in normal situations, it is detected as a loop low performance anomaly by our algorithm.

C. Overall results

Table 7 shows the overall results of anomaly detection on Hadoop and SILK. In the experiments on Hadoop, we detect 15 types of anomalies, 2 of them being false positives (FP). In the experiments on SILK, we detect 91 types of anomalies, 22 of which are FPs. Looking into these FPs, we find that our current loop low performance detection is sensitive to different workloads. This is because the circulation numbers of some loop structures largely depend on the work load. With the help of user's feedback, such FPs can be reduced by relaxing the threshold ϵ.for the corresponding loop structures.

D. Comparison of log key extraction

In order to evaluate our log key extraction method, we compare our method with the method proposed by Jiang et. al. [9]. The comparison results are shown in Table 8, where the numbers of real log key types are manually identified, and are used as the ground truth. For our algorithm, the numbers of obtained log key types are very close to the ground truth. Furthermore, more than 95% of the log keys extracted by our method are identical with the real log keys. By comparison, our algorithm significantly outperforms the algorithm of [9].

Table 7. Overall evaluation results

Anomaly type	Hadoop		SILK	
	Detected anomaly types	False positive	Detected anomaly types	False positive
Work flow error	4	0	16	0
Transition time low performance	6	0	6	0
Loop low performance	5	2	69	22

Table 8. Comparison results of log key extraction

System	Extracted log key types of Jiang et.al [9]	Extracted log key types of our method	Real log key types
Hadoop	257	197	201
SILK	2287	651	631

VIII. CONCLUSION

As the scale and complexity of distributed systems continuously increases, the traditional problem of diagnosis approaches; experienced developers manually checking system logs and exploring problems according to their knowledge becomes inefficient. Therefore, a lot of automatic log analysis techniques have been proposed. However, the task is still very challenging because log messages are usually unstructured free-form text strings and application behaviors are often very complicated.

In this paper, we focus on the log analysis technique for automated problem diagnosis. Our contributions include: (1) We propose a technique to detect anomalies, including work flow errors and low performance, by analyzing unstructured system logs. The technique requires neither additional system instrumentation nor any application specific knowledge. (2) We propose a novel technique to extract log keys from free text messages. Those log keys are the primitives in our model used to represent system behaviors. The limited number of log key types avoids the curse of dimension in the statistic learning procedure. (3) Model the two types of low performance. One is for modeling execution time of state transitions; the other is for modeling the circulation number of loops. In the model, we take into account the factors of heterogeneous environments. (4) The detection algorithm can remove false positive detection of low performance caused by inputting large

workloads. Experimental results on Hadoop and SILK demonstrate the power of our proposed technique.

Future research directions include utilizing log parameter information to conduct further analysis, performing analysis on parallel logs that are produced by multi-thread or event based systems, visualizing the models and the anomalies detection results to give intuitive explanation for human operators, and designing a user-friendly interface.

IX. REFERENCES

[1] W. Dickinson, D. Leon, and A. Podgurski, "Finding Failures by Cluster Analysis of Execution Profiles. In the proceeding of the 23rd International Conference on Software Engineering, May, 2001.

[2] A.V. Mirgorodskiy, N. Maruyama, and B.P. Miller, "Problem Diagnosis in Large-Scale Computing Environments", In the Proceedings of the ACM/IEEE SC 2006 Conference, Nov. 2006.

[3] W. Xu, L. Huang, A. Fox, D. Patterson, and M. Jordan, "Mining Console Logs for Large-Scale System Problem Detection", In Workshop on Tackling Computer Problems with Machine Learning Techniques, Dec. 2008.

[4] C. Yuan, N. Lao, J.R. Wen, J. Li, Z. Zhang, Y.M. Wang, and W. Y. Ma, "Automated Known Problem Diagnosis with Event Traces", In the proceeding of EuroSys 2006, Apr. 2006.

[5] D. Cotroneo, R. Pietrantuono, L. Mariani, and F. Pastore, "Investigation of Failure causes in work-load driven reliability testing", In the proceeding of the 4th International Workshop on Software Quality Assurance, Sep. 2007.

[6] S. Orlando and S. Russo, "Java Virtual Machine Monitoring for Dependability Benchmarking", In proceedings of the 9th IEEE International Symposium on Object and Component-oriented Real –time Distributed Computing, Apr. 2006.

[7] J. Tan, X. Pan, S. Kavulya, R. Gandhi, and P. Narasimhan, "SALSA: Analyzing Logs as State Machines", In the proceeding of 1st USENIX Workshop on the Analysis of System Logs, Dec. 2008.

[8] G. Jiang, H. Chen, C. Ungureanu, and K. Yoshihira. "Multi-resolution Abnormal Trace Detection Using Varied-length N-grams and Automata", in the proceeding of 2nd International Conference on Autonomic Computing, Jun. 2005.

[9] Z. M. Jiang, A. E. Hassa, P. Flora, and G. Hamann, "Abstracting Execution Logs to Execution Events for Enterprise Applications", in the proceeding of the 8th International Conference on Quality Software (QSIC), pp.181-186, 2008.

[10] G. Ammons, R. Bodik, and J. R. Larus, "Mining Specifications", in the proceeding of ACM Symposium on Principles of Programming Languages (POPL), Portland, Jan. 2002.

[11] L. Mariani and M. Pezz`e, "Dynamic Detection of COTS Components Incompatibility", IEEE Software, pp. 76-85, vol.5, 2007.

[12] D. Lo, and S.-C. Khoo, "QUARK: Empirical Assessment of Automaton-based Specification Miners", in proceeding of the 13th Working Conference on Reverse Engineering (WCRE'06), 2006.

[13] Hadoop. http://hadoop.apache.org/core.

[14] J. Dean and S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters", In the proceeding of USENIX Symposium on Operating Systems Design and Implementation (OSDI), Dec. 2004.

[15] S. Ghemawat and S. Leung, "The Google File System", In the proceeding of ACM Symposium on Operating Systems Principles (SOSP), Oct. 2003.

[16] M. Isard, M. Budiu, Y. Yu, A. Birrell, and D. Fetterly, "Dryad: Distributed Data-Parallel Programs from Sequential Building Blocks", In the proceeding of EuroSys, Mar. 2007.

[17] N. Palatin, A. Leizarowitz, A. Schuster, and R. Wolff, "Mining for Misconfigured Machines in Grid Systems", in Proceeding of 12th ACM International Conference on Knowledge Discovery and Data Mining (KDD), pp. 687-692, Philadelphia, PA, USA, 2006.

[18] M. Chen, A.X. Zheng, J. Lloyd, M. I. Jordan, E. Brewer, "Failure Diagnosis Using Decision Trees", in the processing of the first International Conference on Autonomic Computing (ICAC), pp. 36-43, 2004.

JOURNAL OF EMERGING TECHNOLOGIES IN ACCOUNTING
Vol. 1
2004

Principles of Analytic Monitoring for Continuous Assurance

Miklos A. Vasarhelyi
Michael G. Alles
Alexander Kogan
Rutgers, The State University of New Jersey

ABSTRACT: The advent of new enabling technologies and the surge in corporate scandals has combined to increase the supply, the demand, and the development of enabling technologies for a new system of continuous assurance and measurement. This paper positions continuous assurance (CA) as a methodology for the analytic monitoring of corporate business processes, taking advantage of the automation and integration of business processes brought about by information technologies. Continuous analytic monitoring-based assurance will change the objectives, timing, processes, tools, and outcomes of the assurance process.

The objectives of assurance will expand to encompass a wide set of qualitative and quantitative management reports. The nature of this assurance will be closer to supervisory activities and will involve intensive interchange with more of the firm's stakeholders than just its shareholders. The timing of the audit process will be very close to the event, automated, and will conform to the natural life cycle of the underlying business processes. The processes of assurance will change dramatically to being meta-supervisory in nature, intrusive with the potential of process interruption, and focusing on very different forms of evidential matter than the traditional audit. The tools of the audit will expand considerably with the emergence of major forms of new auditing methods relying heavily on an integrated set of automated information technology (IT) and analytical tools. These will include automatic confirmations (confirmatory extranets), control tags (transparent tagging) tools, continuity equations, and time-series cross-sectional analytics. Finally, the outcomes of the continuous assurance process will entail an expanded set of assurances, evergreen opinions, some future assurances, some improvement on control processes (through incorporating CA tests), and some improved data integrity.

A continuous audit is a methodology that enables independent auditors to provide written assurance on a subject matter, for which an entity's management is responsible, using a series of auditors' reports issued virtually simultaneously with, or a short period of time after, the occurrence of events underlying the subject matter.

—CICA/AICPA Research Study on Continuous Auditing (1999)

Companies must disclose certain information on a current basis.

—Corporate and Auditing Accountability, Responsibility, and Transparency (Sarbanes-Oxley) Act (2002)

Please address comments to Miklos Vasarhelyi at miklosv@andromeda.rutgers.edu.

The Guest Editor for this article was Professor Dan O'Leary of the University of Southern California.

Corresponding author: Miklos Vasarhelyi
Email: miklosv@andromeda.rutgers.edu

INTRODUCTION

With the post-Enron support of *continuous assurance* (CA) by the SEC, the AICPA, and Congress, interest in CA has finally reached critical mass. Several years of academic research and conferences culminated in the simultaneous establishment of centers for continuous audit research in the United States and the European Union in September 2002. Three papers in a special issue on CA in the March 2002 volume of *Auditing: a Journal of Practice & Theory* (Alles et al. 2002; Elliott 2001; Rezaee et al. 2002), focused on clarifying the distinction between CA and current audit practices and describing the potential for new assurance products. With CA having been firmly established as the future of auditing, it is now time to shift the focus of the discussion from the potential and promise of CA to a systematic examination of the emerging CA-enabled audit environment.

Vasarhelyi and Halper (1991) predicted that "[Continuous Process Auditing] will change the nature of evidence, timing, procedures, and effort involved in audit work." This paper will first examine the reasons for continuous assurance and then delineate the changes in the nature of assurance encompassing its: (1) objectives, (2) levels and hierarchy, (3) timing, (4) process, (5) tools, and (6) outcomes. The object of this paper is to analyze these changes and the resulting new continuous assurance-enabled audit environment.

This new continuous analytic monitoring-based assurance environment is an outcome of a fundamental transformation in business operations and control: the electronization of the firm through the continued used of legacy systems and the progressive widespread use of Enterprise Resource Planning (ERP) systems. The unique and unprecedented characteristic of ERP is that it seamlessly integrates and automates business processes to achieve real-time information flows. Since CA is progressively being built upon the firm's underlying ERP system, CA inherits these characteristics. However, CA only achieves its full power when it takes full advantage of this ability to automate business processes and integrate information flows. On the other hand, analytic monitoring allows for the increased understanding and monitoring of the integrated and nonintegrated portions of the IT environment. We argue in this paper that the full scope of the capability that automation and integration provides CA has not been fully appreciated and utilized, and show how it provides auditors with an unprecedented toolset that transforms auditing into continuous analytic monitoring of business processes.

While continuous assurance is clearly still an emerging field, the broad forces that will shape its evolution and the nature of the assurance that it will provide are now coming into focus. By identifying the underlying principles of the analytic processes of continuous assurance, and the automation and integration of business processes that give CA its power, our objective in this paper is to provide researchers and practitioners with a clearer roadmap as to how CA is to be implemented, what its capabilities are, and how they are brought about.

We first turn to an examination of the supply and demand for CA services, showing that both have now reached critical mass, driving the recent endorsement of CA by the SEC and the AICPA. Then, in the ensuing sections, we examine the changes in the objectives, timing, processes, tools, and outcomes of the continuous assurance process. The last section provides some concluding remarks.

SUPPLY AND DEMAND FOR CONTINUOUS ASSURANCE

Alles et al. (2002) examined the role of demand in the emergence of CA, and suggested that the major constraint on its adoption was from the demand side, not the supply of the necessary technology. The recent corporate scandals and the passage of the Sarbanes-Oxley Act have only enhanced the demand side effects on CA, removing some of the doubts about its widespread adoption. Elliot (2002), Vasarhelyi (2002), and others have discussed the enabling technologies of CA, such as the use of embedded audit modules (Groomer and Murthy 1989). The new CA-enabled audit environment will emerge from the intersection of these changes in demand, supply, and technology.

The relative speed of expansion or change of each of the three co-determinants affects the feasible set of deployment. Examining these forces in detail, we take the demand-side first. The basic reasons for the need of assurance have only been exacerbated in the new economy, with organizations that are more complex, with more rapid and integrated business processes, and a wider set of legislation and regulations. Many types of management and control information needs exist apart from those served by the traditional financial statement audit, and in the real-time economy these needs can only be satisfied by continuous assurance. In particular, the current series of crises as well as the increasing reliance on technologically enabled business processes suggest new needs for assurance concerning (1) changes in the environment and industry, (2) the existence and effectiveness of controls, (3) increased human resource risks, (4) increased use of outsourced processes, (5) process continuity and integrity, and (6) coherence between endogenous and exogenous factors:

Environment and industry: Over the years, defaults epidemics had plagued particular industries, usually caused by basic economic changes in their environment and by the lack of ability of their management to cope with these changes. This phenomenon has happened in the savings and loans industry and more recently in the telecom industry. After the ensuing wave of defaults and bankruptcies, an intensive set of legal procedures and accusations of improprieties followed. These dramatic changes in the environment are often preceded or occur simultaneously with an increase in the number of mentions in the press and other forms of exogenous indicators. Vasarhelyi and Peng (1999) developed a methodology of semantic parsing and analysis that can serve as an early-warning system for auditors that major environmental changes are occurring with particular clients and more intensive scrutiny is required.

Controls: A key paradigm change in modern business systems concerns the nature of controls. While traditional systems have over the years relied extensively on controls (Vasarhelyi 1980), the intrinsic nature of controls is rapidly changing with automation and the prevalence of IT-based systems often based on ERP systems. Controls in modern systems are typically computer based and entail complex sets of analytics. This requires assurance concerning the existence of controls: that these controls are operational, that their warnings are properly observed and distributed, and that the controls are comprehensive, covering all relevant aspects of operational risk.

Human resources: Major corporate personnel changes serve as a red flag for potential problems and system instability. Templates can be used to look for fraudulent patterns and HR databases can be scrutinized for unorthodox changes. Patterns in personnel changes can indicate problem areas and increased risks.

Outsourcing: The increased outsourcing of business processes is creating virtual parts of businesses that do not naturally flow through the corporation's value chain. New methodologies such as along and across the value-chain analytic monitoring, as well as transaction control tag monitoring, must be used to preserve and evaluate process integrity.

Process integrity: Traditional audit technology has not been able to provide logical links among the pieces of business to define its logical functioning. Intrinsic relationships exist between the parts of business that can be analytically examined, relationally modeled, to give assurance of macro-process integrity.[1]

Internal and external process coherence (integrity): Most organizations operating in a particular industry tend to have a coherent set of operating statistics with operating ratios falling within a predictable range. This allows auditors to define outliers that require examination. The real-time economy now offers a much larger set of dynamic reference points, measurements, and standards. These are obtained through relentless measurement, exponential increase in sensors, intensive collection of statistics, and the progressive adoption of mutually accepted methods of measurements and standards.

[1] See the "Continuity Equations" section later in the paper.

OBJECTIVES OF CONTINUOUS ASSURANCE AND ANALYTIC MONITORING

The basic objective of the traditional audit focuses on providing assurance on the accuracy of the financial statement. Trade-offs between the benefits of this assurance, and the then current information technology led to the development of a materiality threshold of acceptable error. The modern audit, with great improvements in information technology, has changed these trade-offs in the direction of a much finer and timely assurance effort. Eventually, with the increased granularity of data distribution, through the distribution of tagged XML elements, data-level assurance will become necessary. The continuous audit will aim at providing prompter, and more accurate assurance on more granular data for a much wider set of financial and nonfinancial variables.

Levels of Assurance and Audit Objectives

The *audit objectives*—the specific assertions whose verification is the intent of the audit tasks—vary in a continuum, from well-defined issues such as transaction verification, to tasks that are of much higher order of complexity, relying extensively on human judgment, such as the estimation of contingent liabilities. Tasks that are routine and mechanical in nature can be readily transferred from a manual to a CA system and done more comprehensively and cost effectively, taking advantage of the automation and integration of the firm's ERP systems. The question is whether the effectiveness of CA declines monotonically from one end of the audit objective continuum to the other. If that is indeed the case, then the impact of CA on auditing and its ability to create a new audit environment is lessened, as it essentially does not do much more than automate existing audit methods. CA still adds a great deal of value by freeing auditors from mechanical tasks that are better handled by automated systems, thereby giving them more time to focus on matters that require pure human judgment. However, that is still a second- rather than first-order effect on the audit process.

To examine this matter, we propose to distinguish between four levels on the audit objective continuum and examine the role of CA on each one. These four levels of continuous auditing are hard to define in mutually exclusive or exhaustive ways, but they do serve to illustrate the necessary functional dependence of CA on the audit objective. Our four levels of analysis are:

Level 1: Verifying atomic elements of transactions (e.g., movement of money, information, at the data level).

Level 2: Assuring the appropriateness of the measurement rules used in transaction processing (i.e., GAAP).

Level 3: Verifying the adequacy of estimates and their assumptions, as well as the consistency of high-level measurements.

Level 4: Auditing and questioning high-level judgments and facts about the organization.

Exhibit 1 displays in a summary form the four levels of continuous audit, their objectives, procedures, level of automation, and changing paradigms.

While the automation of the first level seems sufficiently straightforward, the really surprising effect of the CA methodology is in its applicability to the other (higher levels). While the extent of application of CA decreases with the increase in the complexity of the audit objective, we argue that certain audit procedures can still be applied, sometimes formalized, and automated even at the high end of the continuum of audit objectives. The key is to undertake formal process mapping, analysis, and reengineering of audit processes. Analogous to the reengineering preceding ERP, it is likely to be the case that a good proportion of audit tasks currently thought to be matters of pure human judgment can, in fact, be systematized to a far greater extent than is currently imagined. The move toward CA

EXHIBIT 1
Levels and Characteristics of Analytic Monitoring

	Level 1 Transactional Verification	Level 2 Compliance Verification	Level 3 Estimate Verification	Level 4 Judgment Verification
Procedures	Rule/waterfall review of data	Formalization of standard relationship with XML derivative	Upstream/down-stream verification	
	Process interruption	Continuity equations	Continuity equations	Continuity equations
	Value chain transaction tracking	Structural knowledge	Value chain relationships	Expert systems
Degree of automation	High	Mixed	Mixed	Low
New paradigms procs., techns.	Continuous reconciliations	Continuity equations	Continuity equations	Continuity equations
	Invisible tracking/ transparent markers		Extensive use of exogenous data	Use of exogenous data
	Automatic confirmations			
	Rule-based trans. evaluation			
		Time-series/cross-sectional analysis	Time-series/cross-sectional analysis	Time-series/cross-sectional analysis

will require auditors to explicitly state the assumptions underlying their estimates and judgments, which is the first step toward bringing these tasks to within the capability of automated CA systems.

We shall now examine in more detail the characteristics of each of the four prescribed levels.

Level 1: Transaction Evaluation

As transactions flow through corporate systems they will be examined, classified, and aggregated, and records of these tasks will be stored by the system at varying points, locations, and degrees of detail. Different types of analysis can be used for different kinds of transactions depending on the type of data they contain. The traditional differentiation between master and transaction data is being progressively refined into a hierarchy of data and storage types depending on factors such as the nature and frequency of the data usage, the geography of the data flow, the location of the activity, the nature of their security and privacy, and existing best practices captured in ERP systems.

Detecting transaction irregularities will range in methods from traditional transaction edits to rule-based evaluations. Basic entry edits include validation of account numbers, checks against lists of clients, regions, products and departments, plausible validity ranges, time validity ranges, and so forth. The validity of these tests depends on the accuracy of various thresholds and other parameters used. The setting of such parameters will typically be done as configuration of a CA system, which has to be reexamined and updated on a regular basis.

Additional verification procedures have to validate the flow of a transaction to make sure that the sequence of processing corresponds to the process specifications defined in the system. Examples of process-flow verifications include checking if the sale corresponds to an inventory movement, to a

bill issued, or to purchase queries received through the website. Real-time process-flow verification becomes possible in CA due to the automation and integration of audit procedures. These verification procedures cannot be done in real time during conventional audits, and very often are not done at all since there is no tight integration of audit processes such as with the audits of accounts receivable and of finished goods inventory.

The continuity and completeness of transactions can be verified in CA using the formal specification of workflow of business processes stored in corporate ERP systems. Automated CA procedures can verify that the transaction has been processed at all the previous steps as required by the process specification. Moreover, structural knowledge of workflow, captured in continuity equations, allows the prediction, to some degree, of transaction flow and whether transactions are missing or have been tampered with. For example, Hume et al. (2000) tapped a very large AT&T biller at many points and succeeded in tracking hundreds of millions of transactions and reconciling their transaction flow. Structural workflow knowledge adds to this reconciliation by allowing flow prediction and loss diagnostics. Note that manual verification of continuity and completeness of a significant number of individual transaction flows presents an insurmountable challenge.

Transaction flow verification within the boundaries of the enterprise, as described above, can be extended beyond these boundaries across the supply chain links if CA is implemented at both ends of a value chain link. This is implemented as a real-time automated confirmation process that creates a certain level of integration between CA systems that are implemented and operated by different assurance providers. Both CA systems will benefit since they can confirm in real time that a receivable booked by Company A matches a payable booked by Company B.

Modern security technology such as encryption and digital signatures can be incorporated in the CA system to prevent or detect transaction tampering. Furthermore, certain types of fraudulent activities have distinct formal patterns and can be detected by matching transactions against fraudulent pattern templates or by using other artificial intelligence techniques such as neural networks (which are currently successfully used for identifying fraudulent credit card transactions).

Level 2: Measurement Rule Assurance (Compliance)

A major task in any audit is to verify that the measurement rules (such as GAAP) are properly applied to the business transactions verified at the first level. Examples of verifying proper rule application include establishing that a certain transaction is properly recorded as revenues, that another transaction is indeed a loan and not a forward contract, or that an expense is properly classified as a capital expense—all examples that have arisen in the current crop of corporate scandals. The problem with automating the verification of such rules in a CA system stems from the fact that while automated rules are strictly formal, the existing rules have a significant amount of imprecision in their formulations. On one hand, if the measurement rules are fuzzy, then they give too much manipulation leeway to the management and cannot be verified. On the other hand, the complexity and variety of modern business transactions make the creation of an exhaustive set of specific measurement rules impractical. The difficulty of finding an appropriate trade-off currently manifests itself in the ongoing extensive debate about principle- versus rule-based accounting standards. Depending on the outcome of this debate, the degree of automation of Level 2 CA procedures will differ.

The automation of CA procedures at this level will utilize a formalization of many measurement rules using knowledge representation methods and the use of automated reasoning techniques. The appropriate technology has been developed in the domain of artificial intelligence and expert systems. Fisher (2003) has demonstrated the feasibility of increased formalization of accounting standards and the benefits of this process. Without going into details of knowledge representation schemes, we can say that a measurement rule is formalized as a special template (whether this

template is a sentence in a first-order language, a Horn clause, or a frame is a matter for another discussion). The hierarchical structure of the accounting standards will be reflected in the formalization so that the templates representing more specific rules override the templates representing more general rules.

The Level 2 procedures will use pattern matching and other techniques to verify an application of rules and either will automatically conclude that this application is justified or will identify this case as unresolved and submit it for the consideration by the human auditors. While the latter cases cannot be guaranteed to be assured in real time, the selectivity of the process will make sure that the scarce resource of human judgment is utilized in the most efficient way. Thus, the participation of human auditors in this type of CA processes is effectively an application of "audit by exception."

Level 3: Estimate Assurance and Consistency of Aggregate Measures

Many estimates are utilized in business measurement and reporting for various reasons. Certain accounting numbers have to be estimated because the underlying information technology made their direct measurement either impossible or too expensive. For example, percentage of work completion used to be difficult to measure, and therefore had to be estimated. However, modern ERP systems and cost accounting techniques allow sufficiently precise measurement of the percentage of work completion in many cases. Note that the fuzziness of accounting standards discussed above may have a direct implication on the difficulty of direct measurement of the percentage of work completion.

A more substantial reason for using accounting estimates is due to the impossibility of knowing the future. Clearly not every account receivable will be collected and not every loan will be paid off. It is usually implicitly assumed that only a human expert can estimate, say, a bad debt allowance. However, many such estimates do not have to be based on intuition. Very often, the intuition of human experts can be captured and formalized in a model that utilizes both internal parameters (like past experience with collecting accounts receivable) as well as external parameters (such as market interest rates, unemployment levels, various economic growth indicators, etc.). The ubiquity of Internet connections to external sources of relevant data and the high level of automation and integration of the firm's own ERP systems make such automatic estimates feasible. Formal models providing such estimates can be incorporated into both ERP and CA systems. Even if a company does not generate an estimate automatically, the CA system can still utilize its own formal model of an estimate to assure in real time that the estimate used by the company is acceptable. Of course, creating a formal model of an accounting estimate is not a simple proposition and may add significant costs to the development of a CA system. A cheaper alternative will be if a company utilizes a formal model for automatically deriving an estimate.[2] Then auditor's task will be reduced to verifying the acceptability of this model, which has to be done only once, and can be done offline, on the basis of whether the parameter values used in the model are reasonable. This is a much simpler task and one that can be automated more readily. While not every estimate can be derived in a formal way, even partial implementation of estimate assurance in the CA system will greatly expand the scope of real-time assurance and reduce the workload on human auditors.

The spectrum of procedures applied at this level of CA includes automatic versions of various analytical review procedures, which will be based not only on internal but also external parameters, which the CA system can receive as an online feed. For example, the distribution representing the aging of accounts receivable can be automatically compared with the distribution derived from the experience of other companies in the industry. If there is a significant discrepancy between the two distributions, or the company has significantly changed the parameters of its estimates, then the CA system can generate an alarm to draw the attention of human auditors. The wide use of automatic

[2] An extreme view of this suggestion may entail that GAAP contain a series of "approved" estimate models, placed in a web library, and corporations use these models disclosing the parameters applied.

managing the pension portfolio. Some of these judgments may be relevant for a wider set of assurance and management services that may eventually arise.

TIMING OF CONTINUOUS ASSURANCE

Online/real-time systems provide the opportunity of immediate assurance processes either simultaneously or just after a particular economic event. This form of verification is different from the pure *ex post facto* nature of the traditional audit process. It provides the opportunity of controlling a process simultaneously or just after the event and in certain cases the ability to interfere with the conclusion of the event correcting its nature. These factors are very different from the traditional audit and should be stated objectively and eventually carefully researched.

A continuous audit procedure, that in CA for example implies day-to-day repetition of an audit step (say reconciliation) becomes a type of meta-control and will eventually become part of a corporation's internal controls. The continuous auditor will then assume the role of secondary verifier by checking if the procedure is really being performed.

A continuous audit procedure that points out an erroneous transaction, and an auditor who acts to correct this error, becomes a proactive actor in corporate information processing. New methods must be developed to maintain his/her independence.

The continuous audit is distributed across the year, performed mainly automatically, and will be a form of "audit-by-exception" where the system is considered materially correct (has an evergreen opinion) until an alarm states it otherwise. The conceptualization of the time frame of a "clean opinion," the meaning of an alarm in the impairment of an opinion, and its usage as audit evidence are further issues for research and the development of standards and principles of practice.

Furthermore, corporate processes have a time cycle of their own. There are instantaneous, hourly, daily, and monthly processes. Each will have a different frame of time for the calculation of their analytics and for the determination of the meaning of an audit alarm.

THE NEW PROCESSES OF CONTINUOUS ASSURANCE

CA will fundamentally change the process of assurance and will consist of an overlay of analytic control processes on top of a monitoring architecture. This section discusses the process, hierarchies, the MC layer, and the steps to be followed confronted with traditional methods.

The Process of Analytic Monitoring

Continuous assurance requires two key components: an IT structure for data gathering and an analytic monitoring methodology to support monitoring, control, and assurance. Since a CA system is an overlay on top of a set of existing systems, the CA IT architecture has to utilize a middleware layer to provide integration between loosely coupled applications such as the firm's ERP system, their legacy systems, and the new web-facing systems. Exhibit 2 shows the proposed architecture of the corporate enterprise systems, where the CA system is shown as an instantiation of the monitoring and control (MC) system.

The system of analytic monitoring uses the MC layer with Key Performance Indicators (KPIs) and formal inter-process relationship models for measurements of flows and levels and to detect variances through metrics and to generate alarms when the standard for discrepancy is reached. This level of analytic monitoring lays on top of a level of actual direct measurement of systems that can be tapped and monitored, as well as processes that still do not have automation and have to rely on pure, high-level analytic monitoring. Clearly, if there are too many discontinuities without direct process monitoring the job of high-level monitoring becomes close to untenable.

TOOLS FOR ANALYTIC MONITORING IN CONTINUOUS ASSURANCE

Each CA level has its own requirements to achieve assurance and, hence, uses different tools and methodologies. As discussed in the first part of this paper, demand is likely to drive CA away from *ex post* evaluation to a closer-to-the-event review. Further, software, people, and analytic thresholds may, at a certain point, intervene into processes and cause their interruption prior to completion. This is a paradigm shift in the nature of auditing that will cause major behavioral resistance and potentially require changes both in the view of independence as well as in many regulations of the professional conduct of accountants. In this more active role, the auditor is part of a meta-control and this intervention process will have to be understood and regulated. To distinguish from the traditional auditor role we call this *analytic monitoring* whereby the functions of performance evaluation, review, assurance, and intervention are rebalanced between auditors, managers, and operational staff.

Understanding both the new demands for assurance and, on the supply side, the automation and integration that underlies CA systems, enables the construction of new audit tools and processes that provide the unique analytic monitoring capability of CA. These new assurance technologies, which are discussed in greater detail by Vasarhelyi et al. (2003), will create an entirely new audit environment. These new technologies facilitate new objectives, processes, and tests, with modern IT systems facilitating a series of intrusive and increasingly transparent activities by analytic monitors:

- Observing events when they happen
- Alarming when exceptions occur
- Drilling down to finer degree of aggregation
- Integrating data across multiple and distinct processes
- Performing repeated tests with low variable cost

We next examine some of the tools that will underlie analytic monitoring in CA and the forces that will shape those tools.

Continuity Equations

The CA environment facilitates bringing an entirely new set of data into assurance processes, with consequently expanded new analytic methods and insights. One category of such analytic methods is what we call *Continuity Equations*, which incorporates structural knowledge into business assurance processes. The objective is to add context to financial data by relating business processes and their ensuing measurements. Structural information about business processes is used to model how data varies with management decisions and how it migrates from process to process throughout the value chain.

The first application of continuity equations was in a tool that was prototyped at Bell Labs in the early 1990s. Exhibit 3 displays a set of sequential processes that entailed bill preparation in the former Bell system (now AT&T).

Transactional data were received from the operating telephone companies in the form of magnetic tapes, which were then extracted into datasets and segmented into other types of datasets that separated types of transactions, which were then rated (priced) and accumulated into 20 different billing cycles. At the end of the cycle at bill pull time, these were rated again now with the optional calling plans that depended on monthly usage for establishing the rates. Finally the bill was prepared, printed, distributed (mailed), and consequently payments started coming in, followed by accounts receivable management, collection actions, customer support, secondary sales, etc. These processes are structurally and logically linked and structural equations provide the model and methodology to use this knowledge in assurance (and management). For example, the effect of a new advertising campaign can be traced through to its impact on usage, billing, and cash receipts. In examining transactions, understanding of knowledge structures serves to identify major breaks in control and

Organizations, when opening bank accounts, signing supplier contracts, or adding vendors to their list of approved vendors, will enclose a mutual confirmation clause and potentially some standard confirmation protocol that will be used by both parties in their mutual transactions. This protocol will present some form of security and code for the type of confirmation obtained (e.g., data-level confirmation, account aggregate confirmation) and this information will be added as a label in the data's XML derivative representation. Automatic procedures in the data flow will review the existence (or lack thereof) of the automatic confirmation and provide summaries and exceptions for the assurance process summaries.

The usage of automatic confirmations will substantially change the nature, procedures, scope, and weight attributed to audit evidence. Confirmations, obtained automatically and highly complemented by self-correcting procedures, will eventually be the most important form of audit evidence. Automatic confirmations, provisioned by extranet agreements, will substantively resolve the audit objectives of existence, completeness, and, to a certain degree, accuracy at the transaction level and account aggregation levels.

Control Tags

Tagging also allows for the inclusion of *control* tags, of which users may or may not be aware, and that can contain sequential numbers, confirmatory information, structural information, data-level assurance measures, and path markings. These tags, aimed at providing auditor information, will also substantially change the weighting of audit evidence, allowing for physical validation of audit objectives. For example, a transaction may have tags with the time of its inception, the time of its passage through key control points, an intelligent sequence number, a prevision of a processing path, and conditions for transaction acceptance and rejection. A control tag may link an order to its payment or other transaction it generated along the value chain. Furthermore, the transaction may leave behind *trailing tags* at the process structural points for transaction validation, alternate path routing, or bot-based verification.

These tools of analytic monitoring, all of which are built upon and take advantage of the automation and integration of the underling IT-enabled business processes, will fundamentally change the way in which auditing will be carried out. The audit environment within which the tools will be used will depend on the changes that CA will also drive on the levels, hierarchy, and process of auditing.

OUTCOMES OF THE CONTINUOUS ASSURANCE PROCESS

Finally, the outcomes of the continuous assurance process will entail an expanded set of assurances, evergreen opinions, some future assurances, some improvement on control processes (though incorporating CA tests), and some improved data integrity.

An Expanded Set of Assurances

The CA module in the MC layer can be programmed to issue both periodic audit opinions as well as current audit opinions, which are updated in real time whenever a change in the situation requires an update. Moreover, opinions of different nature, and alarms of different type, can be issued to different stakeholders such as banks, insurers, federal authorities, state authorities, employee unions, and environmental protection organizations. Such reports (with financial and nonfinancial information) can be tailored to the needs of the stakeholder in question (e.g., asset reports for insurers, environmental reports for OSHA, traditional reports for individual shareholders, etc.). Assurance reports with estimated levels of data reliability can be issued to the stakeholders upon the payment of a fully disclosed fee, set in advance, and open to all entities of that category.

The most important innovation of an audit opinion generated by CA is its explicit futurity, i.e., the promise to continue monitoring and evaluating the operations of the firm, informing the (registered/paying) users/stakeholders if any substantive exceptions occur. The general nature of the evaluative analytics and the magnitude of the limiting variances can be disclosed online, while the auditor could reserve the right to utilize undisclosed models and analytical procedures. The new types of audit opinions provided by CA will result in substantive changes in the timing and role played by assurance in society.

Improvement on Control Processes

The Sarbanes-Oxley Act through its Section 404 requires an auditor's opinion on the quality of corporate internal controls. While the profession is interpreting the law as the requirement to document controls, and their consideration under COSO, the issue of measurement, monitoring, and evaluation of controls in a heterogeneous integrated computer environment is far from being resolved. CA and analytic monitoring can: (1) provide data evidence that controls are functioning without their direct measurement through the understanding of the data consequences of ineffective/nonoperational controls, (2) can repeat computer operation tests (e.g., the test for duplicate payments) activated by auditors to assure that those controls are working, and (3) can "ping" (query) specially designed controls about their operating or pick data from this control on the nature of its functioning.

Improved Data Integrity

The third outcome of analytic monitoring is bringing increased assurance to a lower level of aggregation, in particular the transaction level, providing evidence/inputs for the aforementioned data-level assurance focusing on the tools of Level 1 of analytic monitoring. Automatic confirmations and control tags will provide direct evidence of data-level reliability on particular transactions while high-level monitoring will assure that these are not systematically wrong.

CONCLUSIONS

The progressive electronization of all but the smallest firms has revolutionized the management of business processes and the flow of information within firms. The implementation of ERP results in processes that are automated and integrated to an unprecedented degree, especially since it necessitates that business processes be first reengineered, so bringing them up-to-date and eliminating redundancies and inefficiencies. Continuous assurance systems are built upon a firm's underlying IT systems and so they inherit the ability to rapidly access information from anywhere in the firm's automated and integrated value chain. This will result in fundamental changes in auditing across all its dimensions: objectives, levels and hierarchy, timing, process, tools, and outcomes.

The experience with the evolution of new technologies and business processes suggests that CA will initially be used to do no more than automate existing audit procedures and thereby take full advantage of the capabilities that it has in the new ERP-based environment. This paper describes the tools that will come forward once CA moves to the second stage of its evolution when audit processes are reengineered to exploit the underlying technological capabilities to the fullest. This will lead to the creation of a new system of continuous analytic monitoring that will completely transform the audit environment, in much the same way that ERP systems themselves revolutionized firms' internal monitoring and control systems.

However, to reach that stage will require more than technology implementation. For one thing, it will necessitate auditors actually examining their processes to see if they are susceptible to process mapping and reengineering. This is particularly important if CA is to achieve its full potential, by being progressively extended to higher levels of audit objects, rather than being restricted to the most

mechanical of audit task at the transaction level. However, systematizing processes, once thought to be exclusively in the pure human judgment domain, will take a paradigm shift by auditors. At the same time, continuous analytic monitoring will intrude into the internal control arena, especially since it is built on the firm's own ERP systems. This will create concerns with independence and the relationship between internal and external auditing, analogous to the current debate on the boundary between auditing and consulting.

These implementation issues, the specification of the analytic monitoring toolset and the nature of continuous auditing at each level of audit object all require an intensive research effort, extending from establishing solid theoretical foundations to rigorous laboratory testing. The research agenda put forward by Kogan et al. (1999) needs to be expanded from CA alone to the nature of the entire CA-enabled analytic monitoring environment. While the theoretical work in CA has made progress, the field has been hindered by the lack of a proper set of experimental and empirical research. Consequently, establishing viable data laboratories with large quantity of real (not necessarily but preferably current) data emulating corporate ERP systems, legacy systems, web-facing systems, and real economic circumstances including accounting malfeasance is a priority for the continued development of CA.

REFERENCES

Alles, M. A., A. Kogan, A., and M. A. Vasarhelyi. 2002. Feasibility and economics of continuous assurance. *Auditing: A Journal of Practice & Theory* (Spring).

———, ———, and ———. 2003a. Black box logging and tertiary monitoring of continuous assurance systems. *Information Systems Control Journal* 1: 37–39.

———, ———, and ———. 2003b. Should auditing be completely separated from consulting in the post-Enron era? Working paper, Rutgers, The State University of New Jersey, Rutgers Business School.

American Institute of Certified Public Accountants (AICPA). 1997. *Report of the Special Committee on Assurance Services*. New York, NY: AICPA.

Bovee, M., M. Kogan, K. Nelson, R. P. Srivastava, and M. A. Vasarhelyi. 2002. Financial reporting and auditing agent with net knowledge (FRAANK) and eXtensible business reporting language (XBRL). Working paper, University of Kansas.

Canadian Institute of Chartered Accountants and American Institute of Certified Public Accountants (CICA/AICPA). 1999. *Continuous Auditing*. Research Report. Toronto, Canada: CICA.

Debreceny, R., G. Gray, W. L. Tham, Y. Goh, and P. L. Tang. 2002. The development of embedded audit modules to support continuous monitoring of the control environment. Working paper, Nanyang Technological University, Singapore.

Elliott, R. K. 2001. 21st century assurance. Presentation to the American Accounting Association Auditing Section Midyear Meeting, January 12.

———. 2002. 21st century assurance. *Auditing: A Journal of Practice & Theory* (Spring).

Fisher, I. E. 2003. On the structure of financial accounting standards to support digital representation, storage and retrieval. *Journal of Emerging Technologies in Accounting* 1 (1).

Groomer, S. M., and U. S. Murthy. 1989. Continuous auditing of database applications: An embedded audit module approach. *Journal of Information Systems* 3 (2): 53–69.

Hammer, M. 1990. Reengineering work: Don't automate, obliterate! *Harvard Business Review* (May–June).

Hoitash, R. 2003. Information transfer in analytical procedures: A simulated industry knowledge-management approach. Dissertation draft, Rutgers, The State University of New Jersey, Rutgers Business School, Newark, NJ.

Hume, A., S. Daniels, and A. MacLellan. 2000. GECKO: Tracking a very large billing system. Proceedings of 2000 USENIX Annual Technical Conference, San Diego, California, June 18–23.

Kogan, A., E. F. Sudit, and M. A. Vasarhelyi. 1999. Continuous online auditing: A program of research. *Journal of Information Systems* 13 (2): 87–103.

Research Streams in Continuous Audit: A Review and Analysis of the Existing Literature

Carol E. Brown, Oregon State University, Carol.Brown@bus.oregonstate.edu
Jeffrey A. Wong, University of Nevada at Reno, wongj@comcast.net
Amelia A. Baldwin, University of Alabama in Huntsville, Amelia.Baldwin@uah.edu

Abstract: Advances in information technology and web-based applications are making monitoring and control of operations through continuous auditing increasingly important. The objective of our paper is to summarize and provide a framework for classifying the contributions of the diverse literature addressing the topic of continuous audit. Research streams are divided into five major categories: demand factors, theory and guidance, enabling technologies, applications, and impacts. Over sixty papers have been identified that relate to these areas. Many more articles exist especially in the area of enabling technologies. However, the focus of our paper is the literature most closely related to continuous audit.

I. Introduction

Continuous auditing is based on multiple research streams. Researchers have investigated theories behind continuous auditing, developed concepts of continuous auditing, built a framework for developing continuous audit, and have put together a substantial research agenda. Research in continuous audit has provided substantial guidance for implementation of continuous auditing, particularly in studies supported by professional organizations. Considerable research has been performed to develop the enabling technologies that make continuous auditing possible. Several continuous auditing applications have been described. Researchers have investigated the cost-benefit dynamic and examined the environmental influences that provide impetus and impede development of continuous auditing. Each of these areas will be discussed in the following sections.

Continuous auditing, continuous assurance, continuous monitoring, real-time auditing, and a variety of other terms are used to describe work in this area. Sometimes the terms are used interchangeably and sometimes authors make distinctions regarding which particular activities are described by the different terms. For purposes of this paper we have not attempted to differentiate among the terms. Accordingly, we make no distinctions about the frequency with which auditing procedures are deployed or the intervals of time which are covered. Kogan, et al (1999) provide a short historical perspective on continuous auditing.

Our review of the literature is arranged as follows: section II discusses why continuous audit is in demand, section III describes continuous auditing theory and guidance, section IV discusses various enabling technologies that support continuous auditing, section V addresses continuous audit applications, section VI describes issues related to impacts, and section VII provides concluding comments and ideas for future work.

II. The Demand for Continuous Audit

The demand for continuous audit has arisen from several sources, and information technology has advanced to the point where it seems feasible for this demand to be met. Demand is primarily derived from the users' needs for more relevant or reliable information and changes in the regulatory environment.

External Disclosure

The auditing process has been transformed by the way companies process their transactions and store their data. The audit process has progressed from a largely manual operation to becoming increasingly computer-based (Rezaee, et al., 2002). The next step in computer-based auditing is the continuous audit process. The speed and timeliness offered by continuous auditing practices offers benefits to users of financial information. One of the characteristics of relevant financial information is that it is presented on a timely basis. Increasing the frequency of disclosure improves the timeliness of financial information.

More timely disclosure of financial information is being driven by users' needs, and advances in technology have made more frequent releases of financial information possible. Rezaee et al (2002) assert that real-time financial reporting is a driving force behind the demand for continuous audit. Elliott (2002) indicates that increasing the frequency of disclosure for continuous reporting will drive changes to audit procedures to assure the reliability of the disclosures. The authors predict that the information technology advances that allow for more frequent reporting will enable continuous auditing and that one possible advantage of reporting reliable information on a continuous basis is reducing the cost of capital because of a richer disclosure environment. One study supports the usefulness of more frequent reporting to decision makers. Hunton, et al (2002) conducted an experiment with varying intervals of financial reporting and found that more frequent reporting enhanced the usefulness of information for decision making. Additionally, more frequent disclosures improved the quality of earnings, reduced management's aggressiveness with discretionary accruals and reduced stock price volatility. Hunton, et al (2002) also found that adding auditor assurance to the frequency of reporting resulted in more pronounced effects among participants in the experiment.

Table 1: The Demand for Continuous Audit

External Disclosure
Electronization of Business
Vasarhelyi and Greenstein 2003: Underlying Principles of the Electronization of Business: A Research Agenda
Drives need for CA
More Frequent Disclosure
Huton et al 2002: Assessing the Impact of More Frequent External Financial Statement Reporting and Independent Auditor Assurance on Quality of Earnings and Stock Market Effects
Improved the quality of earnings, reduced management aggressiveness and stock price volatility
Elliott 2002: Twenty-First Century Assurance
Increasing the frequency of disclosure will drive the nature of audit procedures
Rezaee et al 2002: Continuous Auditing: Building Automated Auditing Capability
Real-time reporting, Computer-based IT, Timeliness of reporting
More Timely Detection of Abnormalities
Krass 2002: The Never-Ending Audit: Can Software Prevent Future Enrons?
Not capable of preventing Enron or WorldCom type failure
Vasarhelyi et al 2002: Would Continuous Auditing Have Prevented The Enron Mess?
Enron's special purpose entity abnormalities
Internal Focus
Assure integrity of the transactions
Greenstein and Ray 2002: Holistic, Continuous Assurance Integration: e-Business Opportunities and Challenges
Electronic Commerce
Vasarhelyi et al 2004: Principles of Analytic Monitoring for Continuous Assurance
Outsourcing and EDI
Van Decker 2004: The Need for Continuous Controls Monitoring
Outsourcing and EDI
Laws and Regulations
Other Regulation
Harrison 2005: Embracing Compliance with Continuous Online Auditing
Federal regulations require enforceable and auditable controls
Sarbanes-Oxley
Vasarhelyi et al 2004: Principles of Analytic Monitoring for Continuous Assurance
Section 404 compliance
White 2005: Does Internal Control Enhance or Impede
Important monitoring device that enhances management control
Means and Warren 2005: Continuous Financial Controls Review Processes (CFCRP) . . .
Sections 302 and 404 compliance
Technology
Audit Data Warehouse
Rezaee et al 2002: Continuous Auditing: Building Automated Auditing Capability
Illustrates the functional interrelationships between the audit data warehouse and other users/providers
Electronic Transactions
Kogan et al 1999: Continuous Online Auditing: A Program of Research
Rapid growth of online retailing, securities trading, and procurement systems
ERP environment
Vasarhelyi et al 2004: Principles of Analytic Monitoring for Continuous Assurance
Take advantage of the automation and integration

More timely detection of abnormalities within business and accounting processes is a distinct benefit of continuous auditing. Vasarhelyi, et al (2002) speculates that the abnormal nature of Enron's special-purpose entity accounting would have come to the attention of external auditors sooner had continuous auditing procedures been in place. However, Krasss (2002) speculates that continuous auditing would not be capable of detecting fraud perpetrated at a high level and would not be able to stop the next Enron or WorldCom type accounting-related failure.

Internal focus – controls over day-to-day operations

Many businesses face a complex web of information processing and exchange. Firm-wide information linkages have become more prevalent with the advent of integrated enterprise information systems. The operating environment for firms that are interlinked with others as members of supply chains has become a complex set of relationships which require that data be exchanged between firms. As firms partner with other entities via outsourcing and engage in the exchange of information or transaction processing data through electronic data interchange and other means, the need for scrutinizing the integrity of the transactions is very important (Van Decker, 2004. Vasarhelyi, et al, 2004). Greenstein and Ray (2002) emphasize the importance of assuring integrity in operational data generated by an increasingly electronic commerce environment. Managerial decision models depend upon qualitative and non-financial data generated from closely related (e.g. vendors and customers) and more general sources (e.g. industry benchmarks) in addition to traditional financial data.

Laws and Regulation

Legislative efforts to improve financial reporting accuracy and transparency in the wake of highly publicized corporate accounting failures culminated in the Sarbanes-Oxley act of 2002. The act intensified the focus of both corporate managers and external auditors on internal controls over the processing of information used to produce financial reports. Management is now responsible for assessing the effectiveness of their internal control structure, including any shortcomings. External auditors are required to attest to the accuracy of management's assertion that internal accounting controls are in place, operational, and effective. The continuous monitoring of transactions can assist managers, internal auditors, and external auditors with discovering errors, defalcations, and other breaches of the internal control system in complex, data-intensive environments.

Means and Warren (2005) assert that the complexity of information systems and processes require audit procedures beyond traditional approaches that stress sampling and the presence of internal control procedures. A methodology for continuous financial controls review is suggested by the authors in their paper as one promising way to cope with the demands of the Sarbanes-Oxley act. Vasarhelyi, et. al (2004) assert that continuous auditing

and analytic monitoring techniques may assist Section 404 compliance by (1) providing evidence that controls are functioning and providing an understanding of the consequences of ineffective or non-operational controls, (2) can repeat data operations to assure controls are working, and (3) can query specially designed controls to assure that they are operating. Since continuous auditing and analytic monitoring are able to bring increased assurance to lower level of aggregation at the transaction level, data integrity and reliability are higher, which can reduce risk and the amount of additional work needed for internal and external audits.

Continuous auditing is seen as being an important asset to comply with the Sarbanes-Oxley act, from the perspective of internal auditors. White (2005) asserts that continuous auditing is an important monitoring device that enhances management control of an entity.

Harrison (2005) believes that continuous auditing techniques are the only way to achieve compliance with governing Federal regulations requiring enforceable and auditable controls that will satisfy federal auditors.

Technology-Related

Kogan, et al (1999) state that the rapid growth of online retailing, securities trading, and procurement systems has fueled the need for continuous online assurance. The Elliott Committee (AICPA Special Committee on Assurance Services, 1997) provided further impetus for continuous audit by suggesting that continuous oversight is needed to deal with the substantial changes in societal, economic, and technological developments.

Vasarhelyi, et. al (2004) theorize that continuous auditing is uniquely able to take advantage of information in an enterprise resource planning (ERP) environment. ERP seamlessly integrates and automates certain business processes to achieve real-time information flows. Given the need for real time data, continuous auditing achieves its full potential only when being built on ERP systems that automate business processes and integrate information flows. Computer assisted auditing techniques (CAATS) have limitations because they do not take advantage of the automation and integration of brought about by ERP.

In contrast to Vasarhelyi, et. al (2004), Rezaee et al (2002) suggests that ERP is not a necessary condition, but an audit data warehouse is. The authors illustrate the functional interrelationships between the audit data warehouse and other users/providers of data, including business unit managers, vendors, business end users, and audit end-users.

III. Continuous Auditing Theory and Guidance

Researchers have explored the theory of continuous auditing and have provided substantial guidance for implementation of continuous auditing systems.

Table 2 provides a summary of the major contributions to continuous audit theory and guidance.

Table 2: Theory/Guidance

Guidance

Concepts

Vasarhelyi 2002: Concepts in Continuous Assurance
Outlines continuous assurance concepts

Effects on Audit Process

Rezaee et al 2002: Continuous Auditing: Building Automated Auditing Capability
Relevance Reliability, Internal control

Identify Research Opportunity

Vasarhelyi and Greenstein 2003: Underlying Principles of the Electronization of Business: A Research Agenda
Development of monitoring devices and related alarms that support continuous assurance and the assessment of their relative effectiveness.

Reviews SAS 94 guidance

Rezaee et al 2002: Continuous Auditing: Building Automated Auditing Capability
Effects of IT on internal control structure, types of IT controls, and reporting process under real-time accounting

Framework

Generalized Approach

Vasarhelyi et al 2004: Principles of Analytic Monitoring for Continuous Assurance
Four Levels of Analysis: 1 transaction evaluation, 2 measurement rule assurance, estimate assurance and 3 consistency of aggregate measures, and 4 judgment assurance
Hierarchy of audit processes: Primary, internal controls - activities compared to benchmarks; secondary, external auditing; tertiary, independent review of audit processes

Rezaee et al 2002: Continuous Auditing: Building Automated Auditing Capability
Integrates data sources, an audit data server, and audit data users
Corporate data system, data download, data conversion, audit data warehouse, data transformation, audit workstations

Woodroof and Searcy 2001b: Continuous Audit: Model Development and Implementation within a Debt Covenant Compliance Domain
Components: (1) the various interconnected Web servers; (2) the continuous audit environment; (3) the continuous audit agreement; (4) the characteristics of a reliable system; (5) the characteristics of a secure system; and (6) the evergreen reports.

Techniques

Continuous Financial Controls Review Process

Means and Warren 2005: Continuous Financial Controls Review Processes (CFCRP) . . .
Methodology suggested - "A+++" software – All of the data, fully Audited, All the time

Embedded Audit Modules

Groomer and Murthy 1989: Continuous Auditing of Database Applications: An Embedded Audit Module Approach
Demonstrates approach, advantages and disadvantages discussed

Five classes of tools

Vasarhelyi et al 2004: Principles of Analytic Monitoring for Continuous Assurance
Continuity equations, transaction tagging, time-series and cross-sectional statistical analyses, automatic confirmation, and control tags.

Query Based

Borthick et al 2001: Developing Database Query Proficiency: Assuring Compliance for Responses to Web Site Referrals
Teaching case demonstrating queries for continuous audit

System Performance

Murthy 2004: An Analysis of the Effects of Continuous Monitoring Controls on e-Commerce System Performance
Performance implications of alternative categories of controls

Continuous auditing requires two components:

- an information technology structure for data processing and storage, and
- some type of analytic monitoring methodology to support the assurance function.

Vasarhelyi, et. al (2004) propose an general architecture for continuous auditing that integrates a firm's ERP system, legacy systems, and monitoring and control systems.

Framework for continuous auditing

Vasarhelyi, et. al (2004) develop a theoretical framework for continuous audit that relates the levels of assurance and audit objectives to describe four different levels of analysis in a financial audit. The different levels are characterized by their audit objectives, procedures, level of automation, and paradigms used for their analysis. Each successive level, starting with level one, involves more complexity and judgment than the next. By stratifying the types of assurance functions into levels of complexity or judgment, Vasarhelyi, et. al provide an approach to continuous auditing that attempts to achieve the best match between techniques and audit objectives. The levels in the model are described in the next four paragraphs.

Level 1, *transaction evaluation*, includes the frequent and ongoing transaction flow through corporate information systems. Automated continuous auditing procedures can verify steps taken in transaction processing, and structural knowledge of workflow allows the integrity of transactions processed to be analyzed. Transaction flow verification has implications for extension beyond the borders of an enterprise to supply chain linkages with business partners.

Level 2, *measurement rule assurance*, relates to determining the degree of correspondence between the information and some established criteria such as Generally Accepted Accounting Principles (GAAP). Since many rules within GAAP are "grey" and involve some degree of judgment, simple rule-based continuous auditing techniques may not be able to perform the task of verification.

Level 3, *estimate assurance and consistency of aggregate measures*, encompasses such estimates as warranty expenses and the allowance for bad debts. While such measures may depend on human intuition and judgment, they may sometimes be incorporated into continuous auditing systems if human expertise can be captured and formalized into a model. While such models may be difficult to derive in full, partial implementation of estimate assurance can reduce the workload on human auditors and possibly expand the scope of real-time assurance.

Level 4, *judgment assurance*, relates to the most complex and high-level judgments essential for the future of the organization. The degree of automation at this level is limited. However, continuous auditing systems may be of assistance by providing exogenous data and high-level analysis that will improve the quality of judgments and in turn reduce audit risk. This level requires the most judgment and may require the largest degree of human intervention. Vasarhelyi, et. al (2004) provide an example of the four levels of continuous auditing using pension accounting.

Along with the framework for continuous auditing, Vasarhelyi, et. al (2004) describe a hierarchy of auditing processes comprising primary, secondary, and tertiary processes driven by the degree of connectivity to management and control activities.

- The primary monitoring and controlling process relates to internal control processes whereby enterprise activities are recorded and measured against performance benchmarks to assess how well key activities are meeting the expectations of management.
- The secondary monitoring process takes the perspective of the external auditing function by an independent entity.
- Tertiary monitoring is performed in part by the independent auditor, and in part by another trusted independent party in circumstances such as peer review by another accounting firm or the Public Company Accounting Oversight Board.

Rezaee et al (2002) provide a generalized continuous auditing approach that integrates data sources, an audit data server, and audit data users. Rezaee et al (2002) cite Statement of Auditing Standards (SAS) 94[1] as a source of authoritative literature in auditing that provides some guidance on the effects of information technology and internal control structures important to continuous auditing of a financial reporting process. In particular, SAS 94 provides guidelines for auditors to better understand (1) the effects of information technology on the internal control structure, (2) types of information technology controls that are important to continuous auditing, and (3) the financial reporting process under real-time accounting systems.

Techniques for continuous auditing

Vasarhelyi, et. Al (2004) identified tools to provide new continuous auditing assurance technologies. These tools help to monitor systems largely by detecting variances or exceptions to systems norms as they occur and allowing these exceptions to be investigated to find the root causes. One common consideration with these tools is that they permit continuous auditing at a low incremental cost. Techniques and tools for continuous audit outlined by Vasarhelyi, et. Al (2004) fall into several categories and are briefly described next.

- Continuity equations use business process knowledge and related performance measures to evaluate the reasonableness of actual transactional information. One of the challenges of continuity equations is determining the measurements that accurately and quickly reflect the actions in business processes.
- Transaction tagging allows the transaction flow from one application to the next to be evaluated for data accuracy and integrity. A continuous auditing environment could employ tags that identify the source, nature, assuror and transformation of data. Data level assurance is an important function but presents a challenge in that it is an intricate process and data flows are massive.
- Time-series and cross-sectional statistical analyses are useful in developing models to compare against actual results. The availability of continuous flows of information may make

[1] "The Effect of Information Technology on the Auditor's Consideration of Internal Control in a Financial Statement Audit", April 2001. Amends SAS No. 55 (1998)

time-series regressions more useful than in the past when monthly or quarterly data have been used.

- Automatic confirmations made possible through extranets can substantially increase the amount of audit evidence supporting the existence, value, and other assertions about transactions. Vasarhelyi, et. Al (2004) believe that automatic confirmation procedures have the potential to change the nature, scope, and procedures of an audit because of their ability to fulfill audit objectives at the transaction level.
- Control tags that can contain a range of information can help mark data paths or serve other audit purposes to provide assurance about transaction processing. Such tagging procedures have audit implications for both substantive and control testing. In light of the increased testing of controls required by the Sarbanes Oxley act, control tags may play an important role to cost-effectively provide assurance about the system of internal controls.

Rezaee et al 2002 provide an approach for building continuous audit capacity. Their approach downloads information from the corporate data system, converts it to an appropriate format to store in a audit data warehouse from which it can be used directly at audit workstations or transformed and stored in a data mart for use.

Groomer and Murthy (1989) demonstrate the use of an embedded audit module (EAM) approach in a database environment to capture information about exceptions and violations to the defined data access restrictions. They note that an EAM may have a negative impact on the performance of the application, may generate large data sets and are at risk of being modified by application programmers. They note that with an EAM: violation and exception information can be captured on a real time basis; all events can be screened, not just a sample; and the extent of compliance testing may be reduced. To make an EAM approach viable, auditors must be knowledgeable about application and database environment, have the client cooperation, and have a stable hardware and software environment in which to implement.

Murthy (2004) examines the effects of adding continuous auditing processing loads to overall system processing capacity. The authors analyze the system performance implications that alternative categories of continuous auditing monitoring controls have on web-based e-commerce applications under varying system load conditions. This paper provides a methodology that can be used by potential implementers of continuous auditing to investigate the effects of incorporating various monitoring controls into their application systems. Three categories of controls were examined, ranging from relatively simple controls involving only calculations to complicated controls requiring the use of structured query language aggregate functions. It was found that aggregate function controls had very detrimental effects on system performance even under light system load conditions. Capacity planning is an essential consideration for information technology system architects.

IV. Enabling Technology

Several enabling technologies have been identified for continuous auditing.

These include: belief functions, databases, expert systems, intelligent agents, neural networks, real time accounting and XBRL/XML. How each of these areas relates to continuous auditing is discussed in more depth below. Highlights of relevant research in each of these areas are summarized in Tables 3a through 3g.

Belief Functions:

The belief function framework is an evidential reasoning approach for mathematically combining evidence for and against an assertion. It differs from probabilistic methodologies in that lack of evidence is not interpreted as either supporting or invalidating the assertion. Belief functions have been demonstrated to provide a good method for aggregating audit evidence (Gillett and Srivastava 2000; Shafer and Srivastava 1990; Srivastava and Mock 2000; and Srivastava and Shafer 1992 and Sun et al 2006). Sun et al (2006) describes real world implementation of belief functions for information system security risk analysis from the perspective of the auditor. The environments in which continuous audit systems must operate are complex, having multiple types of interrelated transactions, objectives, vulnerabilities, and controls. The belief function framework provides structured yet tractable approach to risk assessment in complex environments, making it an approach that may prove useful for continuous audit.

Table 3a: Enabling Technology

Belief Functions
Assurance Services for Electronic Commerce
Srivastava and Mock 2000: Evidential Reasoning for WebTrust Assurance Services Evidential network (belief function) model and decision-theoretic model for WebTrust assurance
Audit Risk
Srivastava and Shafer 1992: Belief-Function Formulas for Audit Risk Relates belief functions to the structure of audit risk
Compare with Bayesian
Shafer and Srivastava 1990: The Bayesian and Belief-Function Formalisms: A General Perspective for Auditing Compares the Bayesian formalism with the belief-function formalism
Information System Security Risk
Sun et al 2006: An Information Systems Security Risk Assessment Model under the Dempster-Shafer Theory of Belief Functions Information system security risk analysis using belief functions
Integrating Sampling Results
Gillett and Srivastava 2000: Attribute Sampling: A Belief-Function Approach to Statistical Audit Evidence How statistical evidence obtained by means of attribute sampling may be represented as belief functions

Databases and data analysis technology:

Business processes have become increasingly reliant on electronic processing and storage. Simultaneously, the tools for enabling continuous auditing methods have manifested themselves along with technological advances. This section discusses the technological factors that are a necessary condition for continuous audit.

Kogan, et al (1999) state that continuous auditing is only feasible if implemented as a fully automated process

and one that allows instant access to relevant events and their outcomes. Electronic data, plus the dramatic reduction in the cost of hardware and gains in processing power, have paved the way for continuous auditing. Rezaee et al (2002) suggest that the standardization of data sources, particularly from legacy systems, will be the most complex and challenging aspect of building continuous auditing capacity. The risks and costs of introducing errors and duplicate records can create huge obstacles to the development of an end-user continuous auditing, testing, and analysis systems.

Table 3b: Enabling Technology

Databases
Data Source
Rezaee et al 2002: Continuous Auditing: Building Automated Auditing Capability List necessary conditions; Scalable audit data warehouse
Murthy and Groomer 2004: A Continuous Auditing Web Services Model for XML-Based Accounting List necessary conditions
Kogan et al 1999: Continuous Online Auditing: A Program of Research Fully automated process that allows instant access to relevant events and their outcomes
Database Queries
Borthick et al 2001: Developing Database Query Proficiency: Assuring Compliance for Responses to Web Site Referrals Teaching case demonstrating queries for continuous audit

Rezaee et al (2002) propose a model of continuous auditing that does not require an ERP data warehouse, but does require an audit data mart. Data collected and transformed for data marts of various business units will be physically stored in an audit data server for easy access, analysis, and reporting. The essence is not to duplicate corporate databases, but to selectively collect transactions that have been defined to pose an audit issue and store them in a separate data warehouse. According to Rezaee et al (2002), an integrated audit data mart must have, at minimum, the following characteristics:

- Integrated query, analysis, and reporting through a unified user interface
- An easy-to-use product line, yet powerful enough for the most sophisticated analytical users.
- Capacity to export the results of queries easily to common spreadsheets and database systems.
- A query engine capable of retrieving and processing large volumes of data.
- Data aggregation and multidimensional database capability
- Capabilities for: advanced statistical modeling and data exploration.
- Data visualization capability for data mining exploration and identification of patterns and trends in the data.

Murthy and Groomer (2004) have a similar list of necessary conditions including:

- A highly reliable client system that can provide the necessary information to the auditors on a timely basis.
- The subject of the audit must be electronically accessible.
- The auditor must be proficient in information systems computer technology and what is to be audited.
- Automated procedures must reliably provide the needed audit evidence.
- A highly placed executive must champion continuous auditing.

Borthick et al (2001) developed a teaching case to demonstrate a query-based approach to continuous monitoring. They note that this approach is likely to become more common as trading partners increasingly use networks to directly interact with each others systems.

Expert Systems

Expert systems have been applied to a variety of audit tasks. Expert systems were the first application of AI in auditing, with significant growth in this area beginning in the 1980s (Denna et al 1991, Brown and Murphy 1990, Abdolmohammadi 1987, Hansen and Messier 1987). Successful applications of AI to audit tasks have mostly addressed structured repetitive tasks where the human expertise is not extremely hard to acquire. This research stream has resulted in a number of expert systems in use at public accounting firms, including ADAPT (Gillett 1993), Deloitte Touche's Audit Planning Advisor, Price Waterhouse's Planet, Arthur Andersen's WinProcess and KPMG's KRisk (Zhao et al 2004, Bell et al. 2002, Brown 1991). The computational models developed by those early expert systems researchers have facilitated the automation of the audit process and continuous audit.

Table 3c: Enabling Technology

Expert Systems
Research Review
Denna et al 1991: Development and Application of Expert Systems in Audit Services Evaluate research and development in the design of expert systems for the audit domain
Abdolmohammadi 1987: Decision Support and Expert Systems in Auditing: A Review and Research Audit domain
Hansen and Messier 1987: Expert Systems in Auditing: The State of the Art Literature review
System Descriptions
Brown 1991: Expert Systems in Public Accounting: Current Practice and Future Directions Deloitte Touche's Audit Planning Advisor, Price Waterhouse's Planet, Arthur Andersen's WinProcess
Brown and Murphy 1990: The Use of Auditing Expert Systems in Public Accounting Review of systems in use by big 6
Gillett 1993: Automated Dynamic Audit Programme Tailoring: An Expert System Approach and KPMG's KRisk ADAPT
Zhao et al 2004: Auditing in the E-commerce Era Deloitte Touche's Audit Planning Advisor, Price Waterhouse's Planet, Arthur Andersen's WinProcess and KPMG's KRisk
Bell et al 2002: KRisk: A Computerized Decision Aid for Client Acceptance and Continuance Risk KPMG's KRisk

Intelligent Agents

An intelligent agent is goal-oriented software that uses the internet to seek out information to achieve its specific goal.

The FRAANK (Financial Reporting and Auditing Agent with Net Knowledge) system (Bovee et al 2005, Kogan et al 2002) acquires, parses, and converts financial information available on the WEB to XBRL, which can then be automatically processed to achieve an audit objective. FRAANK's earlier incarnation EDGAR Agent (Nelson et al 2000; Nelson et al 1998) searched for the current cash balance and calculated some financial ratios of a user-specified firm, whether or not the user specified the firm name in exactly the same way as it is filed in the EDGAR database. Woodroof and Searcy (2001, 2001b) demonstrate the use of an intelligent agent to continuously assure debt covenant compliance. These systems demonstrate that intelligent agents can be used in continuous auditing systems to either query on-line databases or to query trading partners for information. This capability can facilitate much richer continuous audit applications, potentially giving the capability of continuously confirming receivables and payables with large trading partners or comparing reported financial information of trading partners and competitors with that of the firm being audited.

Figure 3d: Enabling Technology

Intelligent Agents

Debt Covenant Compliance

Woodroof and Searcy 2001: Audit Implications of Internet Technology: Triggering Agents Over the Web in the Domain of Debt Covenant Compliance
Describes implementation

EDGAR Agent

Nelson et al 1998: Virtual Auditing Agents: The Edgar Agent Example
Quality service/framework; expand audit practice; use in audit

Nelson et al 2000: Virtual Auditing Agents: The EDGAR Agent Challenge
Quality service/framework

FRAANK

Bovee et al 2005: Financial Reporting and Auditing Agent with Net Knowledge (FRAANK) and extensible Business Reporting Language (XBRL)
Describes development and application

Kogan et al 2002: Design and Applications of an Intelligent Financial Reporting and Auditing Agent with Net Knowledge (FRAANK)
Describes prototype

Neural Networks:

A neural network is a type of artificial intelligence system that connects relatively simple processing elements to form a network. The number of processing elements, how they are connected and the weights of the connections determine the output based on specified inputs. A neural network can be created to model any function. Neural networks are particularly effective when a large database of prior examples exists with known inputs and outputs.

Neural networks have been investigated for a variety of audit tasks. Calderon and Cheh (2002) and Koskivaara (2004) provide an in-depth review of audit research using neural network technology. Fadlalla and Lin (2001) look more broadly at financial tasks, including bankruptcy prediction and credit granting among others. Fadlalla and Lin note that application categories suitable for neural networks are "unstructured and data intensive and involve much uncertainty, hidden relationships, and noise," (2001 p.115) which is an apt description of the continuous audit area.

Figure 3e: Enabling Technology

Neural Networks

Analytical Review

Lin et al 2003: A Fuzzy Neural Network for Assessing the Risk of Fraudulent Financial Reporting
Integrated fuzzy neural network (FNN) for fraud detection

Coakley and Brown 1993: Artificial Neural Networks Applied to Ratio Analysis in the Analytical Review Process
Error Detection - seeded material errors in monthly data

Green and Choi 1997: Assessing the Risk of Management Fraud Through Neural Network Technology
Management Fraud Detection

Fanning and Cogger 1998: Neural Network Detection of Management Fraud Using Published Financial Data
Fraud Detection - used published financial data

Koskivaara 2000: Artificial Neural Network Models for Predicting Patterns in Auditing Monthly Balances
Built models using the 72 monthly balances of a manufacturing firm

Fanning et al 1995: Detection of Management Fraud: A Neural Network Approach
Fraud Detection

Control Risk Assessment

Ramamoorti et al 1999: Risk Assessment in Internal Auditing: A Neural Network Approach
Research investigates whether neural networks can help enhance auditors risk assessments

Davis et al 1997: Supporting Complex Audit Judgment Tasks: An Expert Network Approach
Prototype expert network to support internal control risk assessment

Review of Research

Koskivaara 2004: Artificial Neural Networks in Analytical Review Procedures
Overview of artificial neural network (ANN) studies conducted in the auditing field

Fadlalla and Lin 2001: An Analysis of the Applications of Neural Networks in Finance
Financial tasks

Calderon and Cheh 2002: A roadmap for future neural networks research in auditing and risk assessment
Audit neural network

Screening

Baker 2005: Fraud and Artificial Intelligence
Transaction screening: fraud detection and prevention

Viaene et al 2002: A Comparison of State-of-the-Art Classification Techniques for Expert Automobile Insurance Claim Fraud Detection
Claims screening: automobile insurance claim fraud detection

The most relevant individual studies for continuous audit are those investigating control risk assessment (Ramamoorti,et al 1999; Davis et al 1997); and. analytical review including error and fraud detection (Coakley and Brown 1993; Fanning and Cogger, 1998; Fanning et. al. 1995; Green and Choi 1997, Koskivaara 2000, Lin et al 2003). Neural networks have long been used to screen transactions and claims for fraud prevention by banks, credit card companies and the insurance industry (Baker 2005; Viaene et al 2002). The success of neural networks in these areas holds promise for the use of neural networks in continuous audit applications. Although the black box nature of neural networks poses some issues, the ability of neural networks to quickly tag unusual transactions for additional screening could prove useful as a way to minimize the impact of continuous audit on the performance of accounting systems.

Real-time Accounting

Scanners, card readers, databases, enterprise-wide management systems, supply chain management systems and networks have made much of today's transaction processing a real-time activity. At the retail end, transactions are recorded with electronic cash registers that are connected to the company's inventory and accounting records and the credit card and electronic funds transfer systems. EDI and just-in-time inventory systems have brought much of the manufacturing and wholesale accounting to real-time processing as well. Rezaee et al (2000) and Alles et al (2004b) argue that as accounting becomes increasingly real-time, continuous auditing must follow.

Figure 3f: Enabling Technology

Real-time Accounting
Discusses Issues
Alles et al 2004: Restoring Auditor Credibility: Tertiary Monitoring and Logging of Continuous Assurance Systems.
Impact on Auditing
Rezaee et al 2000: Real-Time Accounting Systems Preventitive and detective controls

XBRL/XML

XBRL (eXtensible Business Reporting Language) is an XML (eXtensible Markup Language) based standard for communicating financial information. "The purpose of XBRL is to facilitate the preparation, publishing, exchange and analysis of financial statements and the information they contain. It serves as an open specification for data, developed by a global, non-profit consortium comprising companies, associations, and government agencies" (Baldwin et al 2006). XBRL comprises a standard or specification, a family of taxonomies, instance documents and style sheets.

The use of XBRL allows systems to understand the meaning of the tagged data from the definitions in the taxonomy used. Style sheets can be used to present the information in an appropriate format, whether for a human or computer system reader. The use of XBRL technology can facilitate the development of reusable continuous audit modules as well as gathering relevant data from external sources. The FRAANK system (Bovee et al 2005, Kogan et al 2002) demonstrates how it can be used to facilitate analytical review. Murthy and Groomer (2004) presents a model for continuous auditing web services (CAWS) using XML and web services.

Table 3g: Enabling Technology

XBRL/XML
Analytical Review
Kogan et al 2002: Design and Applications of an Intelligent Financial Reporting and Auditing Agent with Net Knowledge (FRAANK) FRAANK/XBRL
Bovee et al 2005: Financial Reporting and Auditing Agent with Net Knowledge (FRAANK) and eXtensible Business Reporting Language (XBRL) FRAANK/XBRL
Indirect Link to Client Database
Woodroof and Searcy 2001b: Continuous Audit: Model Development and Implementation within a Debt Covenant Compliance Domain XML/XBRL Downside - push rather than pull technology
Monitoring
Murthy and Groomer 2004: A Continuous Auditing Web Services Model for XML-Based Accounting Continuous Monitoring Web Services/XML
Standardization of Data
Rezaee et al 2002: Continuous Auditing: Building Automated Auditing Capability XBRL
Validating Mechanism
Debreceny and Gray 2001: The Production and Use of Semantically Rich Accounting Reports on the Internet: XML and XBRL Validating mechanism for Internet published statements

V. Continuous Audit Applications

Continuous auditing will involve different processes than the traditional audit. Vasarhelyi, et. al (2004) outline a series of steps that begin with a considerable amount of front-end architectural work. Key features of the processing steps outlined by Vasarhelyi, et. al (2004) are the identification of key performance metrics, identification of factors that cause discrepancies between actual results and those that are modeled, and continuous updating of the process models over time.

Glover et al (2000) conducted a survey of 2,700 members of the Institute of Internal Auditors to assess how and to what extent software was being used for their work, including continuous monitoring. Nearly half of the respondents reported that they use some type of continuous monitoring software, which is up from twenty-four percent reported in a similar survey in 1998. The authors surmised that the ability of commercial software to meet the specific needs of internal auditors for continuous monitoring may be increasing.

Rezaee et al (2002) believe that the use of data extraction and analysis software by audit departments has been increasing rapidly and has been supplanting the traditional manual methods of auditing. In their paper, Rezaee et al (2002) note two companies that are using continuous audit applications: Carolina Power and Light, and Exxon Company, USA.

> "Carolina Power and Light has adopted Selective Monitoring and Assessment of Risks and Trends or SMART Auditing to identify potential problems, unfavorable trends, and unusual variances measured by key indicators. Using CATTs, the Audit Services

identify testing criteria, (4) automate tests, (5) communicate test results, (6) receive feedback, and (7) track progress of continuous auditing results. Nelson (2004) walks through the seven steps as they were applied by the implementation team at HCA, Inc.

One interesting application domain for continuous auditing has been described by Turoff et al (2004). The authors contend that continuous auditing techniques can be integrated with the Emergency Response Management Information System to form a more effective and reliable supporting component of homeland security. Other applications for continuous audit are likely to emerge as the use of continuous auditing becomes more widespread.

Sigvaldason and Warren (2005a, 2005b) describe a relatively new continuous auditing software application with the brand name AuSoftware designed to continuously monitor the enterprise systems of large companies and governmental agencies. The software imports data into an audit data mart, as described by Rezaee (2002), that facilitates continuous assessment and analysis with selected algorithms. The data is stored for subsequent analyses such as time series analysis.

VI. Impacts – costs and benefits

The trend towards increasingly affordable computing power may translate to continuous auditing being a cost-effective assurance tool. Means and Warren (2005) state that continuous audit (continuous financial controls review processes) software will automate and replace some manual and visual assurance processes that will provide ongoing efficiencies and savings.

Benefits may be difficult to quantify, but are often linked to two factors. The first is more timely disclosure of accurate financial information to market participants. The second factor is timelier discovery of errors, omissions, and defalcations. Automated, software-driven audit procedures are often more cost-effective than their manual predecessors. Benefits and cost savings accrue to both internal and external monitors (auditors) of the firm.

Rezaee et al (2002) discuss factors related to continuous audit that can help to reduce the costs of monitoring. Costs related to the basic audit assignment may be reduced by allowing auditors to test client's transactions in a manner that is faster and more efficient than manual testing. In addition to the efficiencies of testing, a larger sample (up to 100%) of the population may be tested, potentially increasing the quality of evidence gathered and reducing audit risk. Efficiencies in testing also allow the auditor to spend more time on understanding the client's business/industry and internal control structure within the time constraints allocated to a particular audit.

Searcy and Woodroof (2003b) point out various types of inefficiencies faced by external auditors that continuous auditing can mitigate. Continuous auditing techniques may reduce the time that it takes for the client to provide auditors with data needed to complete their audit tasks. The review process used as a quality control measure during the audit may be automated to shorten the time taken through continuous auditing procedures. In addition, continuous audit can reduce inefficiencies in the audit process and reduce errors and mistakes (Searcy and Woodroof 2003).

Table 5: Impacts – costs and benefits

Benefit

Rezaee et al 2000: Real-Time Accounting Systems
CA increases quality

Database Access Audit

Pathak et al 2005: Minimizing Cost of Continuous Audit: Counting and Time Dependent Strategies
Focus on the long run operating cost of running database audit

Efficiencies and Cost Savings

Means and Warren 2005: Continuous Financial Controls Review Processes (CFCRP) Using Powerful New Technologies may be the Only Real Answer to the Demands of Sarbanes-Oxley
A+++ is far less costly approach that lowers overall risk

Searcy and Woodroof 2003: Continuous Auditing: Leveraging Technology
Reduce time to get information and do procedures

Rezaee et al 2002: Continuous Auditing: Building Automated Auditing Capability
Reduces cost of audit. Reduces amount of time spent on audit.
Specifies testing criteria for auditing throughout the year.
Simultaneously performs substantive and control tests.

Woodroof and Searcy 2001: Audit Implications of Internet Technology: Triggering Agents Over the Web in the Domain of Debt Covenant Compliance
Reduce overall audit staff.

Increased Effectiveness and Efficiency

Searcy and Woodroof 2003: Continuous Auditing: Leveraging Technology
Reduce time delays due to not having needed information, reduce inefficiencies and starts and stops in the audit process, automate and shorten the review process and reduce errors and mistakes.

Reduce Audit Risk

Rezaee et al 2002: Continuous Auditing: Building Automated Auditing Capability
Increases quality of audit, test larger sample or population.

Reduce Cost of Capital

Elliott 2002: Twenty-First Century Assurance
Reduce the cost of capital because of a richer disclosure environment.

Behavioral Impacts

Hunton et al (2002): Assessing the Impact of More Frequent External Financial Statement Reporting and Independent Auditor Assurance on Quality of Earnings and Stock Market Effects.
Behavioral impacts (reduced management's aggressiveness with discretionary accruals and reduced stock price volatility) on users of more frequent disclosures is more pronounced when information is audited

VII. Conclusions and Direction for Future Work

A wide and varied literature exists relating to continuous audit. Substantial work still needs to be done to develop a clear understanding of how the various research streams relate to each other. Additional literature related to the enabling technologies must be identified and how those technologies relate to continuous audit should be clarified.

As business processes become increasingly more interlinked through the use of information technology and web-based applications, continuous auditing will become a more important monitoring and assurance device. Other

factors creating the demand for continuous auditing are the Sarbanes-Oxley act, and the needs of external users for reliable financial disclosures that are released frequently. The question remains, will the demand factors identified actually lead to more continuous audit applications? Will demand factors lead to different models of who pays for the cost of audits?

The existing literature provides a good theoretical framework that provides academics and practitioners with valuable guidance for implementing and maintaining continuous auditing applications. An excellent example is Vasarhelyi et al (2004), which provides a conceptual framework for continuous auditing. While the literature is becoming more developed, work remains to be done in this relatively young discipline.

More descriptions of actual systems are needed along with descriptions of their architecture and the techniques used. Performance data from actual implementations would be helpful. Research on the impacts of continuous audit systems is needed. What are the actual costs to develop and run systems? What are the actual benefits? How do such systems change behaviors?

Two papers assist in providing future direction for research in continuous auditing. Hunton et al (2004) explore opportunities for future work related to the use of continuous auditing procedures. They assert that the advent of continuous auditing will provide behavioral accounting researchers with many topics to study related to how frequency of reporting and assurance affect judgment and decision-making processes. The authors suggest that bounded rationality and cognitive limitations may come into play with the possible overload of available information. Although the article is not a recent one, Kogan, et al (1999) outline a number of areas for research related to continuous auditing that are still relevant and of interest to academics and practitioners.

References

Abdolmohammadi, Mohammad J. 1987. Decision Support and Expert Systems in Auditing: A Review and Research Directions. *Accounting and Business Research* (Spring) 173-185.

Abdolmohammadi, M. J. and Sharbatouglie, A., 2005, *Continuous Auditing: An Operational Model for Internal Auditors*, The Institute of Internal Auditors Research Foundation, Altamonte Springs, Florida, 1-159, http://www.theiia.org/bookstore.cfm?fuseaction=product_detail&order_num=5001.

Alles, Michael; Brennan, Gerard; Kogan, Alexander and Vasarhelyi, Miklos A..2006. Continuous monitoring of business process controls: A pilot implementation of a continuous auditing system at Siemens.*International Journal of Accounting Information Systems*, 7(2), 137-161.

Alles, M., Kogan, A., and Vasarhelyi, M.A., 2002, Feasibility and Economics of Continuous Assurance, *Auditing: A Journal of Practice and Theory*, 21(1), 125 – 138.

Alles, M., Kogan, A., and Vasarhelyi, M. A., 2004a, Restoring Auditor Credibility: Tertiary Monitoring and Logging of Continuous Assurance Systems, *International Journal of Accounting Information System*s, 5(2), 183-202.

Alles, M., Kogan, A., and Vasarhelyi, M.A., 2004b, Real Time Reporting and Assurance: Have Their Time Come?, *ICFAI Reader*, Institute of Chartered Financial Analysts of India, Special issue: Finance in 2004.

Baker, N., 2005, Fraud and Artificial Intelligence, *Internal Auditor*, 61(1), 29-31.

Baldwin, A. A.; Brown, C. E. and Trinkle, B. S. 2006. XBRL: An Impacts Framework and Research Challenge, *Journal of Emerging Technologies in Accounting*, forthcoming.

Bell, Timothy B., Jean C. Bedard, Karla M,. Johnstone and Edward F. Smith. 2002. KRisk: A Computerized Decision Aid for Client Acceptance and Continuance Risk Assessments. *Auditing: A Journal of Practice & Theory* **21**(2:September) 97-113.

Boccasam, P. V. and Kapoor, N., 2003, Managing Separation of Duties Using Continuous Monitoring, *IT Audit*, 6:, The Institute of Internal Auditors, , Available online: http://www.theiia.org/itaudit/index.cfm?fuseaction=forum&fid=5433.

Borthick, A. F.; Jones, D. R. and Kim, R., 2001, Developing Database Query Proficiency: Assuring Compliance for Responses to Web Site Referrals, *Journal of Information Systems*, 15(1), 35-56.

Bovee, M.; Kogan, A.; Nelson, K.; Srivastava, R. P..; Vasarhelyi, M. A. 2005. Financial Reporting and Auditing Agent with Net Knowledge (FRAANK) and extensible Business Reporting Language (XBRL). *Journal of Information Systems*, Spring, 19(1), 19-41.

Brown, Carol E. 1991. Expert Systems in Public Accounting: Current Practice and Future Directions. *Expert Systems with Applications*, **3**(1: June), 3-18.

Brown, Carol E. and Murphy, David S. 1990. The Use of Auditing Expert Systems in Public Accounting. *Journal of Information Systems* 4(2:Fall) 63-72

Calderon, T. G. and Cheh, J. J., 2002, A roadmap for future neural networks research in auditing and risk assessment, *International Journal of Accounting Information Systems*, 3(4), 203-236.

CICA/AICPA, 1999, *Continuous Auditing, Research Report*, The Canadian Institute of Chartered Accountants, Toronto, Canada.

Coakley, J. R. 1995. Using Pattern Analysis Methods to Supplement Attention-Directing Analytical Procedures, *Expert Systems with Applications,* 9(4), 513-528.

Coakley, J. R. and Brown, C. E., 1993, Artificial Neural Networks Applied to Ratio Analysis in the Analytical Review Process, *International Journal of Intelligent Systems in Accounting, Finance and Management*, 2(1), 19-39.

Coderre, David 2006, A Continuous View of Accounts. *Internal Auditor*, April, 63(2), 25-31.

Davis, J. T.; Massey, A. P. and Lovell R. E. R. 1997. Supporting Complex Audit Judgment Tasks: An Expert Network Approach. *European Journal of Operations Research.* 103(2), 350-372.

Debreceny, R., and Gray, G. 2001. The Production and Use of Semantically Rich Accounting Reports on The Internet: XML and XBRL. *International Journal of Accounting Information Systems* 2(1:January) 47-74

Debreceny, R., G. Gray, J. Ng, K. Lee, and W. Yau. 2005. Embedded Auditing Modules in Enterprise Resource Planning

International Journal of Engineering Technology Science and Research
IJETSR
www.ijetsr.com
ISSN 2394 – 3386
Volume 4, Issue 3
March 2017

Study on Google Firebase for Website Development
(The real time database)

Hari Shankar Singh, Uma Shankar Singh
Dept. of Computer Engineering
R.N. Modi Engineering College.
Kota, India

ABSTRACT

The purpose of this study is to introduce everyone to Google Firebase and it's features. Firebase provides database and file storage. APIs provided by firebase are platform independent and synchronize in real time. Firebase has many features like Storage, Authentication, Database, Notifications etc. Nowadays many websites are using firebase to build modules or a complete project. In this article we will cover how to use firebase to build your application.

Keywords—Real time database; APIs (application programming interface); Authentication; quick development;

I. INTRODUCTION

A database is a collection of information that is organized in such a way which is easily manageable and accessible. Firebase is a "NoSQL" database which are useful for large sets of distributed data. NoSQL databases are effective for big data performance issues that relational databases aren't built to solve. Along with this, firebase is also a "Real Time" database which provides an API that allows developers to store and sync data across multiple clients. Data in firebase is saved as json and can be exported. Implementing Firebase is quick and easy. With intuitive APIs packaged into a single SDK, we can focus on quickly building our app and not waste time building a complex infrastructure.

II. FIREBASE FEATURES

A. Built-In Analytics

One of the best features of Firebase is the Analytics dashboard that it comes equipped with. It is free and has the capacity to report 500 event types, each with up to 25 attributes. The dashboard is really top notch for looking at user behavior and measuring various user attributions. Ultimately it helps you understand how people use your app so you can better optimize it in the future.

B. Easy App Development

With Firebase, you can focus your time and attention on developing the best app(s) possible for your business. Since the operations and internal functions are so solid, and taken care of by the Firebase interface, you can spend more time developing the high quality app that users are going to want to use. more reliable way across platforms

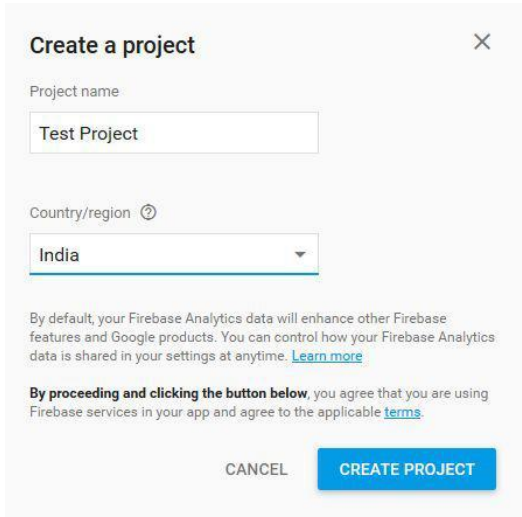
2. **Authentication**: Have a lot less friction with acclaimed authentication
3. **Hosting**: Deliver web content faster
4. **Remote Configuration**: Customize your app on the go
5. **Test Lab**: Test in the lab instead of on your users
6. **Crash Reporting**: Keep your app stable
7. **Realtime Database**: Store and sync app data in realtime
8. **Storage**: File storing made easy

C. Notifications

You can very easily manage notification campaigns, including having the ability to set and schedule messages in order to engage users at the right times of day. These notifications are totally free and unlimited for both Android and iOS. There is only one dashboard to worry about, and if you integrate with Firebase Analytics you can use a variety of user segmentation features.

III. ADDING FIREBASE TO PROJECT (WEB)

First we need to add our project to firebase, for which we should follow the below steps:



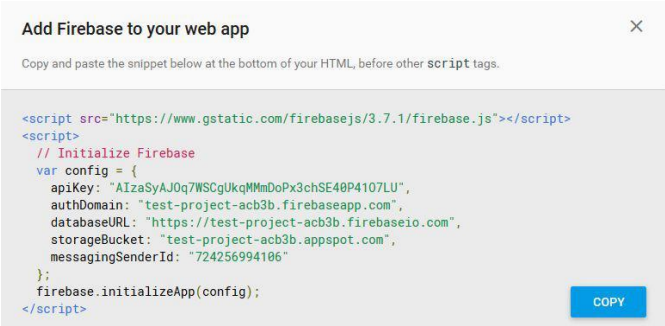
International Journal of Engineering Technology Science and Research
IJETSR
www.ijetsr.com
ISSN 2394 – 3386
Volume 4, Issue 3
March 2017

1. **Cloud Messaging**:Deliver and receive messages in a new Project".
2. Then fill "Project Name" and select "Country / Region" as "India".
3. Now click on "Create Project" which will redirect you to "Project Configuration" page.
4. Click on "Add Firebase to your web app" and copy the "Initialization code" provided in pop-up.
5. Download the firebase "Web SDK".

IV. INITIALIZING FIREBASE SDK

To include firebase code in your project and initialize it's SDK, we follow below steps:

1. Copy the SDK to projects "Libraries" directory.
2. Include "firebase.js" in the footer of view file.
3. Paste the "Initialization code snippet" in the footer, which we copied from "Firebase Console".



4. Firebase has below more components which could be included as per app requirements:

- firebase-app - The core firebase client
- firebase-auth - Firebase Authentication
- firebase-database - The Firebase Realtime Database
- firebase-storage - Cloud Storage
- firebase-messaging - Firebase Cloud Messaging

These components could be added with below javascript code:

<script src="https://www.gstatic.com/firebasejs/3.7.1/firebase-app.js" ></script>

1. Go to "Overview" section and click on "Cloud Messaging"

<script src="https://www.gstatic.com/firebasejs/3.7.1/firebase-messaging.js"></script>

<script src="https://www.gstatic.com/firebasejs/3.7.1/firebase-storag e.js"></script>

V. RETRIEVING AND WRITING DATA

After the basic setup, we begin with the actual coding, which would include "Reading" data from firebase and writing to it.

A. Initialize firebase configuration

We initialize firebase by using below javascript code:

```
<script>
  var config = {
    // ...
  };
  firebase.initializeApp(config);
</script>
```

B. Get a database reference

First we need to get an instance of firebase's database reference "firebase.database.Reference" .

<script src="https://www.gstatic.com/firebasejs/3.7.1/firebase-auth.js "></script>

<script src="https://www.gstatic.com/firebasejs/3.7.1/firebase-databa se.js"></script>

```
// Get a reference to the database service
var database = firebase.database();
```

C. Basic writes operations

We use set function to **write** data at some specific reference or node.Using **set()** function, we overwrite the existing data if any, at the particular node.

```
function writeUserData(userId, name, email, imageUrl) {
  firebase.database().ref('users/' + userId).set({
    username: name,
    email: email,
    profile_picture : imageUrl
  });
}
```

International Journal of Engineering Technology Science and Research
IJETSR
www.ijetsr.com
ISSN 2394 – 3386
Volume 4, Issue 3
March 2017

Using the above code we are writing user's profile information at "users/userId" node.

For **reading** data, we call function **once()** on any database reference from where we want to retrieve data.

```
var userId = firebase.auth().currentUser.uid;
return firebase.database().ref('/users/' + userId)
.once('value').then(function(snapshot) {
  var username = snapshot.val().username;
  // ...
});
```

Here we are retrieving user's data from "users/userId" node

D. Listening to events

Events are triggered when there are some changes in the real time database. We add listener to these changes so that we can manipulate them in real time.

We have many event listeners like "child_added", "child_changed", "child_removed", "child_moved". Whenever there is a change these events are triggered.

E. Sorting data

We use below functions to sort data that comes from firebase once() call.

Method	Usage
`orderByChild()`	Order results by the value of a specified child key.
`orderByKey()`	Order results by child keys.
`orderByValue()`	Order results by child values.

F. Filtering data

When there is need to filter the result set, we use filter functions which can be combined with sort functions as well. Other than below two functions, there are other functions for filtering as well, which are **startAt()**,**endAt()**and **equalTo()**.

Unlike the sorting methods, we can use multiple filter functions in the same call to retrieve more useful data.

Method	Usage
`limitToFirst()`	Sets the maximum number of items to return from the beginning of the ordered list of results.
`limitToLast()`	Sets the maximum number of items to return from the end of the ordered list of results.

CONCLUSION

In this paper, we have studied about google firebase, and it's extremely useful features Firebase is extremely useful and reliable to make real time applications or websites nowaday. A lot of bigger brands are using firebase and it's features for the same.

ACKNOWLEDGMENT

The authors would like to thank Mr. Bhupendra Soni (Vice Principal) for his help.

REFERENCES

[1] Firebase Web Codelag https://codelabs.developers.google.com/codelabs/firebase-web
[2] Firebase Web Documentation https://firebase.google.com/docs/web/setup
[3] Sample Firebase Apps https://firebase.google.com/docs/samples/
[4] Firebase Realtime Database Tutorial https://www.101apps.co.za/index.php/item/182-firebase-realtime-databa se-tutorial.html
[5] Firebase Tutorial: Real-time Chat https://www.raywenderlich.com/140836/firebase-tutorial-real-time-chat-2.
[6] Firebase Php SDK https://github.com/ktamas77/firebase-php
Getting Started with Firebase API Tutorial Using PHP cURL http://www.techplugg.com/firebase-api

Systematically Deriving Quality Metrics for Cloud Computing Systems

Matthias Becker Sebastian Lehrig Steffen Becker
{matthias.becker|sebastian.lehrig|steffen.becker}@uni-paderborn.de
Software Engineering Group & Heinz Nixdorf Institute
University of Paderborn, Paderborn, Germany

ABSTRACT

In cloud computing, software architects develop systems for virtually unlimited resources that cloud providers account on a pay-per-use basis. Elasticity management systems provision these resource autonomously to deal with changing workload. Such changing workloads call for new objective metrics allowing architects to quantify quality properties like scalability, elasticity, and efficiency, e.g., for requirements/SLO engineering and software design analysis. In literature, initial metrics for these properties have been proposed. However, current metrics lack a systematic derivation and assume knowledge of implementation details like resource handling. Therefore, these metrics are inapplicable where such knowledge is unavailable.

To cope with these lacks, this short paper derives metrics for scalability, elasticity, and efficiency properties of cloud computing systems using the goal question metric (GQM) method. Our derivation uses a running example that outlines characteristics of cloud computing systems. Eventually, this example allows us to set up a systematic GQM plan and to derive an initial set of six new metrics. We particularly show that our GQM plan allows to classify existing metrics.

Categories and Subject Descriptors

D.2.8 [**Software Engineering**]: Metrics—*Performance measures*; D.2.11 [**Software Engineering**]: Software Architectures—*Languages*

Keywords

cloud computing, scalability, elasticity, efficiency, metric, SLO, analysis, GQM

1. INTRODUCTION

In cloud computing, software architects develop applications on top of compute environments being offered by cloud providers. For these applications, the amount of offered resources is virtually unlimited while elasticity management systems provision resources autonomously to deal with changing workloads. Furthermore, providers bill provisioned resources on a per-use basis [1]. As a consequence of these characteristics, architects want their applications to use as few resources as possible in order to save money while still maintaining the quality requirements of the system. Quality properties that focus directly on these aspects are scalability, elasticity, and efficiency [9].

These quality properties need to be quantified for requirements engineering and software design analysis by means of suitable metrics. For instance, cloud consumers and cloud providers need to negotiate service level objectives (SLOs), i.e., metrics and associated thresholds [7]. Such SLOs have to consider characteristics like changing workloads ("how fast can an application adapt to a higher workload?") and pay-per-use pricing ("how expensive is serving an additional consumer?"). However, no established metrics for requirements/SLO engineering and software design analysis exist. Current metrics assume knowledge of implementation details as they focus on the application at run-time [9] and lack a systematic derivation making such limitations explicit.

In literature, classical performance-oriented metrics [4] like response time and throughput fail for situations relevant for cloud computing applications. First, they do not take changing workloads into account, e.g., metrics to describe reaction times to system adaptations are missing. Second, the degree to which systems match resource demands to changing workloads cannot be quantified. More recent work [9] proposes initial metrics for such characteristics that assume knowledge of implementation details like resource handling. Therefore, these metrics are inapplicable when such knowledge is unavailable, e.g., in early design phases and for SLOs.

To cope with these lacks, we systematically derive an initial set of scalability, elasticity, and efficiency metrics using the goal question metric (GQM) method [15]. First, we illustrate the characteristics of cloud-aware applications using a running example scenario and, subsequently, derive metric candidates from this scenario. Second, by generalizing our metric candidates, we develop a first set of six metrics. Third, we show that our GQM plan allows to systematically classify existing metrics and makes their limitations explicit.

This short research paper is organized as follows. Section 2 gives definitions of the considered quality properties and Sec. 3 introduced the running example system and its requirements. The system is implemented as cloud-aware application. In Sec. 4, we systematically derive a set of new metrics using the GQM method. In Sec. 5, we put our metrics in relation to existing metric proposals in literature and classify these metrics using our GQM plan in Sec. 6. Finally, Sec. 7 concludes the paper and highlights future work.

ICPE 2015, January 31-February 4, 2015, Austin, TX, USA.

2. DEFINITIONS

Because we distinguish scalability, elasticity, and efficiency throughout our paper, we first give the definitions of these properties based on the work of Herbst et al. [9]: "**Scalability** is the ability of the system to sustain increasing workloads by making use of additional resources" [9]; "**Elasticity** is the degree to which a system is able to adapt to workload changes by provisioning and deprovisioning resources in an autonomous manner, such that at each point in time the available resources match the current demand as closely as possible" [9]; and "**Efficiency** expresses the amount of resources consumed for processing a given amount of work" [9]. We use these definitions throughout this paper to derive metrics for each quality property. In the next section, we exemplify these definitions based on our running example.

3. MOTIVATING EXAMPLE

As an example scenario, we consider a simplified online book shop. An enterprise assigns a software architect to design this shop, given the following requirements:

R_{fct}**: Functionality** In the shop, customers shall be able to browse and order books.

R_{scale}**: Scalability** The enterprise expects an initial customer arrival rate of 100 customers per minute. It further expects that this rate will grow by 12% in the first year, i.e., increase to 112 customers per minute. In the long run, the shop shall therefore be able to handle this increased load without violating other requirements.

R_{elast}**: Elasticity** The enterprise expects that the context for the book shop repeatedly changes over time. For example, it expects that books sell better around Christmas while they sell worse around the holiday season in summer. Therefore, the system shall proactively adapt to anticipated changes of the context, i.e., maintain a response time of 3 seconds or less as well as possible. For non-anticipated changes of the context, e.g., peak workloads, the system shall re-establish a response time of 3 seconds or less within 10 minutes once a requirement violation is detected.

R_{eff}**: Efficiency** The costs for operating the book shop shall only increase (decrease) by $0.01 per hour when the number of customers synchronously using the shop increases (decreases) by 1. In other words, the marginal cost of the enterprise for serving an additional customer shall be $0.01.

Requirements R_{scale}, R_{elast}, and R_{eff} are typical reasons to operate a system in an elastic cloud computing environment [9], i.e., an environment that autonomously provisions the required amount of resources to cope with contextual changes. Thus, the software architect designs the shop as a 3-layer Software as a Service (SaaS) application operating in a rented Infrastructure as a Service (IaaS) cloud computing environment that provides replicable virtual servers (see Fig. 1). The three layers involve the typical layers of web applications: presentation, application, and data layer.

The architect designs each SaaS layer such that it can consume a higher/lower quantity of IaaS services to sustain changing workloads. Technically, he ensures that each SaaS layer can be replicated and load-balanced over additional IaaS virtual servers (scale-out) or be removed again (scale-in). Therefore, properties like scalability (R_{scale}), elasticity (R_{elast}), and efficiency (R_{eff}) of the book shop are inherently coupled with corresponding properties of the underlying IaaS environment. The enterprise (IaaS consumer) and the IaaS provider have, thus, to agree on measurements and thresholds for these properties. Typically, consumer and provider achieve an agreement based on negotiated SLOs as exemplified by R_{scale}, R_{elast}, and R_{eff}. However, currently there is a lack of agreed-on metrics for scalability, elasticity, and efficiency in the context of cloud computing. This lack results in too few SLOs or SLOs that cannot be quantified and checked by cloud consumers. In consequence, there is the risk that requirements R_{scale}, R_{elast}, and R_{eff} cannot be fulfilled due to a non-anticipated (and contractually uncheckable) behavior of the underlying IaaS environment.

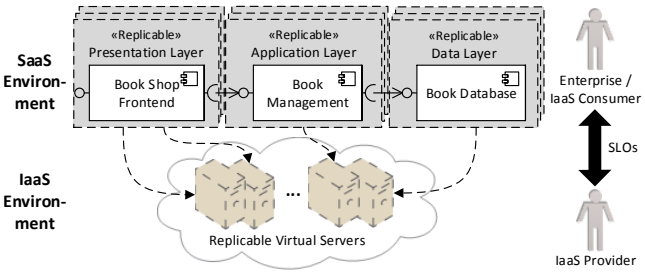


Figure 1: Overview of the simplified online book shop.

4. DERIVING METRICS WITH GQM

We derive our metrics with the goal question metric (GQM) method [15] in a systematic top-down fashion by first defining the goal to analyze cloud computing system designs, e.g., for design analysis or SLO specification (conceptual level). Second, we formulate questions that help achieving the goal (operational level). Finally, we identify metrics that allow us to answer the questions (quantitative level).

4.1 Goal

Table 1 shows the goal in form of the GQM goal definition template. The metrics we want to define in this paper shall help to analyze (purpose) the scalability, elasticity, and efficiency (issue) of cloud computing system designs (object) from the viewpoint of a software architect (viewpoint).

Note that the derived metrics are not necessarily generalizable or applicable for different objects or viewpoints. For example, cloud computing vendors may perceive the efficiency of a cloud system software different than software architects and, thus, need different metrics.

Table 1: Goal definition according to GQM plan

Purpose	Analyze
Issue	the scalability, elasticity, and efficiency of
Object	cloud computing system designs
Viewpoint	from a software architect's viewpoint

4.2 Questions

Next, we define questions that help achieving our defined goal. We define questions for each quality property separately. All questions are indicator questions for the according quality property as defined in Sec. 2.

Questions for Scalability.

According to the definition, scalability is an *ability*, i.e., a system is either scalable or it is not. Hence, we define questions whether a system is able fulfill its requirements under increasing workload. Moreover, the increase rate of the workload can be a relevant context factor.

$Q1_{scale}$ Does the system fulfill its requirements when the workload increases (from workload WL_X to WL_Y)?

$Q2_{scale}$ Does the system fulfill its requirements when the workload increases with rate R (from workload WL_X to WL_Y within time t)?

Questions for Elasticity.

Elasticity is, according to the definition, the degree to which a system is able to autonomously adapt to workload changes. Thus, we define questions that consider the time it takes for the system to adapt.

$Q3_{elast}$ How often does the system violate its requirements under workload WL_X in time period Δt?

$Q4_{elast}$ From a point the system violates its requirements, how long does it take before the system recovers to a state in which its requirements are met again?

Questions for Efficiency.

Efficiency relates the amount of demanded resources to the amount of work requested. Hence, we formulate questions that ask for this relation.

$Q5_{eff}$ How close to the actual resource demand can the system align the resource provisioning?

$Q6_{eff}$ What is the amount of resources, autonomously provisioned by the system, for a given workload WL_X?

4.3 Metrics

In this section, we summarize general requirements for metrics and concrete requirements for cloud computing system metrics. We then derive our metrics that answer the questions from Sec. 4.2 in two steps. In the first step, we derive exemplary metrics (EM) for scalability, elasticity, and efficiency for the example system in Sec. 3. In the second step, we generalize these metrics to answer the questions for arbitrary cloud computing systems.

4.3.1 Requirements for Derived Metrics

The metrics we define have to meet four typical characteristics of metrics [7] in order to be applicable by software architects: (1) *quantifiability*, (2) *repeatability*, (3) *comparability*, and (4) *easy obtainability*. Additionally, we require our metrics to be (5) *context dependent* to reflect the context dependency of cloud computing systems. Figure 2 shows parts of the context that impact the quality of a cloud computing system as a feature diagram. This context covers the system's workload, i.e., work and load, and deployment, i.e., replication of components, processor speed, memory size, network speed, etc. All of this context is subject to change at run-time in a typical cloud computing environment. Hence, metrics need to have a context parameter to enable the comparison of implementations in different contexts. For example, system $\mathcal{S}_A$ may have less SLO violations per minute with workload WL_X (context) but more SLO violations per minute with workload WL_Y (different context) compared to system $\mathcal{S}_B$.

4.3.2 Derived Metrics from Example Scenario

In this section, we define exemplary metrics (EM) that can be used to answer the questions from Sec. 4.2. In this section, we restrict these metrics to evaluate whether the requirements for the example scenario in Sec. 3 are fulfilled. In Sec. 4.3.3, we generalize these metrics for arbitrary cloud computing systems.

The first requirement R_{fct} in our example scenario is a general requirement that defines the basic functionality of the book shop. However, metrics for functional requirements are out of the scope of this paper.

Exemplary Scalability Metrics.

R_{scale} specifies the system's required scalability, i.e., the system's ability to make use of additional resources at increasing workloads. Hence, this requirement is dependent on the context, e.g., the load specified as arrival rates. The book enterprise has to specify arrival rates, e.g., by estimating current sales, sale trends, and seasonal sale variability.

The book shop's software architect checks whether the designed system fulfills requirement R_{scale} by evaluating questions $Q1_{scale}$ and $Q2_{scale}$ for this design. A metric EM_{scale}, defined as the maximum workload the system can handle without violating requirement R_{scale}, answers question $Q1_{scale}$. For example, the software architect can measure this metric by predicting the performance for the book shop design with the increased workload of 12% as expected by the book enterprise (predictions can, e.g., be conducted using queuing networks, c.f. [14]).

The enterprise does not specify at which rate the workload increases, e.g., linearly or exponentially. Therefore, to answer question $Q2_{scale}$, the book shop's software architect needs a metric EM_{rate}, defined as the rate a system can scale up to a certain maximum workload. For example, the software architect can measure this metric by predicting the performance of the book shop design under increasing workload, e.g., a linear increasing workload of one additional customer per month. Afterwards, the architect can discuss the quantified requirement with the enterprise.

Exemplary Elasticity Metrics.

R_{elast} specifies the book shop's required elasticity, i.e., the degree to which the book shop is able to adapt to workload changes by autonomously provisioning and deprovisioning cloud resources. In general, these workload changes (context) can be either anticipated or impossible to anticipate. As described in Sec. 3, the enterprise can anticipate variability of the book shop's customers demand, i.e., the enterprise estimates to sell more books around Christmas than in mid-summer. Other short-term variations of the workload cannot be anticipated, e.g., peak workloads.

The book shop's software architect can evaluate whether R_{elast} is fulfilled by answering questions $Q3_{elast}$ and $Q4_{elast}$ for the book shop design. The cloud computing system can potentially cope with both, anticipated and non-anticipated, workload variability. However, considering the fact that resources can be available with delay, the system will likely violate requirement R_{elast} until the time additionally provisioned resources are available. A metric EM_{viol}, defined as

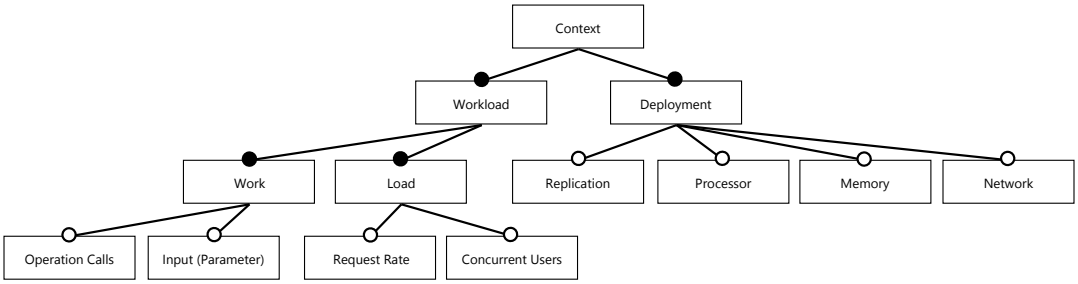


Figure 2: Feature diagram showing aspects of context.

number of requirement violations for a given workload, is a mean for the software architect to answer question $Q3_{elast}$. A metric EM_{adapt}, defined as the time between detection of a requirement violation and the time when the requirement is fulfilled again, e.g., by autonomous resource provisioning, answers question $Q4_{elast}$. For example, the book shop's architect can measure both metrics, EM_{viol} and EM_{adapt}, by simulating the book shop's autonomous behavior using self-adaptive queuing networks [2].

Exemplary Efficiency Metrics.

R_{eff} specifies the required efficiency of the book shop regarding its resource consumption. The book shop's autonomous provisioning and deprovisioning adapts the resource consumption to the book shop's workload, i.e, the workload is important context here as well. Furthermore, the operation costs of the book shop depend on this resource consumption as the enterprise has to pay for the cloud resources in a pay-per-use fashion.

The software architect can evaluate whether the R_{eff} is fulfilled by answering questions $Q5_{eff}$ and $Q6_{eff}$. A metric EM_{close} can be defined as the difference between the book shop's amount of provisioned resources for a workload WL_X and the minimum amount of resources required to cope with that workload WL_X without violating the stated requirements. The book shop's software architect can use this metric to answers question $Q5_{eff}$. A metric EM_{prov}, defined as the amount of resources provisioned for a workload WL_X, answers question $Q6_{eff}$. Whereas the marginal costs can be calculated directly from the metric EM_{prov}, EM_{close} supports the software architect to determine whether the required marginal costs are achievable. Again, both metrics can be measured by simulating the book shop's autonomous behavior under variable workload, for example.

4.3.3 Generalized Metrics

We define initial new metrics that we derived from the six exemplary metrics (EM) from the previous subsection. For each metric, we define the corresponding quality property it quantifies, what is being measured, on which contextual properties the measurement depends, and the scope and unit of the measurement result. To guarantee objectivity, our metrics rely only on externally observable properties of a system, e.g., costs per time. Reproducibility is guaranteed by specifying the contextual properties on which the metric depends. Thus, context dependency is also guaranteed. By defining the measurement result's scope as ordinal numbers, we guarantee that the metric reflects a testable quantification of the quality property. Additionally, we provide a small example for each metric. Table 2 summarizes the following generalized metrics.

Scalability Metrics.

Scalability Range (ScR) Based on EM_{scale}, we define scaling range as a scalability metric that reflects a cloud computing system's ability to achieve its SLOs in a certain workload range, e.g., a range of request rates. For each single workload within this range, the system achieves its SLOs. The workload range is defined as a maximum workload. For example, a perfectly scalable system $\mathcal{S}_A$ can scale up to a infinite request rate, i.e., $ScR_{\mathcal{S}_A} = \infty\ req/min$. A less scalable system, for example, scales up to a request rate of 112 requests per minute, i.e., $ScR_{\mathcal{S}_B} = 112\ req/min$.

Scalability Speed (ScS) Based on EM_{rate}, we define scalability speed ScS as a workload range with a maximal change rate in which a system can scale. This metric is a scalability metric which additionally considers the rate at which a system can scale. That is, the metric defines that a system is able to achieve its SLOs at each time when the workload changes at a maximal changing rate. The rate is defined by a maximum workload and an increase rate. For example, a scalable system can scale up to a request rate of 112 req/min with a linear increase rate of 1 additional requests per month, i.e., $ScS_{\mathcal{S}_A} = (112\,req/min, 1\,req/month)$.

Elasticity Metrics.

Mean Time To Quality Repair ($MTTQR$) We derive the mean time to quality repair metric from EM_{adapt}, the measure how quickly a system can adapt to workload changes. It is a measure for elasticity and depends on a workload delta, i.e., the increase/decrease between two workloads. $MTTQR$ defines the mean time a system needs to reestablish its SLOs when the workload increases/decreases for a defined workload delta specified as factor (real number). Hence, $MTTQR$ is measured in time units. Since it defines a mean time, $MTTQR$ is specific for a specified time frame in which the mean is calculated. For example, with the same workload increase factor of 1.2, a perfectly elastic system will adapt itself to the increasing workload within zero time, i.e., $MTTQR_{\mathcal{S}_A,1d}(1.2\ req/s) = 0\ min$. A less elastic system, e.g., will need a mean time of 10 min (calculated over one day) to adapt itself to increasing workload after it detects the workload increase, i.e., $MTTQR_{\mathcal{S}_B,1d}(1.2\ req/s) = 10\ min$.

Number of SLO violations ($NSLOV$) The number of SLO violations in a defined time interval is derived from EM_{viol}. This metric measures elasticity of a system. The workload delta is specified as a factor (real numer) as well.

Table 2: Derived quality property metrics for cloud computing systems

Metric	Unit	Example
Scalability Metrics		
Scalability Range (ScR)	max	The system scales up to 112 req./min
Scalability Speed (ScS)	(max, rate)	The system scales up to 112 req./min with linear increase rate 1 req./month
Elasticity Metrics		
Number of SLO Violations ($NSLOV$)	1/[time unit]	42 SLO (response time) violations in 1 hour
Mean Time To Quality Repair ($MTTQR$)	[time unit]	30 seconds for an additional 10 requests/hour
Efficiency Metrics		
Resource Provisioning Efficiency (RPE)	$[0, \infty]$	10% more resources than actual resource demand
Marginal Cost (MC)	[monetary unit]	\$1.00 for an additional 100 requests/hour

$NSLOV$ reflects how often a system violates its SLOs when workload changes at a given rate, measured as a real number. For example, with a workload increase factor of 1.2, a perfectly elastic system would have 0 SLO violations per request, i.e., $NSLOV_{\mathcal{S}_A}(1.2\ req/s) = 0$. In contrast, a non-elastic system will violate its SLO for each request, i.e., $NSLOV_{\mathcal{S}_A}(1.2\ req/s) = 1$.

Efficiency Metrics.

Resource Provisioning Efficiency (RPE) We define resource provisioning efficiency (RPE) as a metric to measure a system's efficiency in a specified workload delta based on EM_{close}. That is, the metric measures the mismatch between actual resource utilization and resource demand while the workload is changing. We measure this mismatch in percentage, i.e., as a real number. A perfectly efficient system will adapt its resource demand exactly to the resource demand at all times. For example, if the workload increases with factor 1.2 the system will provision exactly that amount of additional resources required to cope with this additional workload, i.e., $RPE_{\mathcal{S}_A}(1.2\ req/s) = 0$.

Marginal costs (MC) We derive the *marginal costs* for a specified workload delta from EM_{prov}. Marginal costs are the operation costs to serve one additional workload unit, thus, measuring the efficiency of a cloud computing system. For example, the operation costs to serve 20% additional requests per second (factor 1.2) can be \$1.00 for system $\mathcal{S}_A$, i.e., $MC_{\mathcal{S}_A}(1.2) = \$1.00$.

5. RELATED WORK

In this section, we present related work that also targets metrics for scalability, elasticity, and efficiency in the context of cloud computing and SLO specification. Where applicable, we classify found metrics using our GQM plan in Sec. 6.

In the area of **scalability metrics**, Bondi [5] further divides scalability into structural scalability ("ability to expand in a chosen dimension without major modification") and load scalability ("ability of a system to perform gracefully as the offered traffic increases"). In terms of this classification, we focus on load scalability in our work. However, in contrast to Bondi, we also provide concrete metrics for load scalability and apply these to the cloud computing context. Duboc et al. [6] formally define scalability requirements and provide a method to analyze a software model for scalability obstacles. A scalability obstacle could also be defined and detected using our metrics. In contrast to the approach from Duboc et al., our metrics are closer to SLOs in their current form. Jogalekar and Woodside [11] present a scalability metric for general distributed systems. Their scalability metric also includes an efficiency measure. In our work, we distinguish between scalability and efficiency as two different metrics. Furthermore, in contrast to Jogalekar and Woodside, our metrics are focused on cloud computing environments with their particular characteristics.

Herbst et al. [9] provide a set of **elasticity metrics** based on speed and precision (w.r.t. avoiding under- and overprovisioning) of scale-in and -out. Because their goal is a benchmarking methodology for elasticity, they can assume full knowledge about the resource usage of the benchmarked application. However, in our case, we assume that this knowledge is unavailable because details on resources are implementation decisions. In contrast, we consider requirements specified between cloud consumer and provider. This lack of knowledge necessarily leads to different metrics as by Herbst et al., e.g., considering SLO violations instead of resource usage transparent to consumers. Folkers et al. [8] and Islam et al. [10] both provide elasticity metrics that meet this requirement regarding knowledge. However, they lack the distinction between elasticity and efficiency because they both use cost metrics for elasticity, thus, eliminating the possibility to investigate both properties in separation.

Roloff et al. [12] define basic **efficiency metrics** for high performance computing in cloud computing environments. They define cost efficiency as the product of costs per hour and average performance. In contrast to our work, they neglect the context, e.g., actual workload, and only take the average performance. Berl et al. [3] address energy efficiency in all technical components of cloud computing, e.g., servers, networks as well as network protocols. We do not address energy efficiency directly but only resource provisioning efficiency in this paper. Investigating whether efficient provisioning of resources positively correlates with energy efficiency is left as future work.

These related works focus on single quality properties. In contrast, the Cloud Services Measurement Initiative Consortium (CSMIC) provides a standard **measurement framework**, called the Service Measurement Index (SMI), that covers all quality properties considered important for cloud computing [13]. CSMIC particularly provides metrics for these quality properties, intended to be used by cloud consumers and cloud providers. Their framework allows for a structured classification of quality properties, similar to our GQM plan. However, CSMIC does not address the deviation of their suggested metrics, thus, leaving open what limitations come with their metrics and whether other metrics are feasible as well.

6. CLASSIFICATION OF METRICS IN RELATED WORK

In this section, we classify metrics identified in related work (Sec. 5) by relating these metrics to the questions of our GQM plan. In doing so, we show that we can systematically make assumptions and limitations of existing metrics explicit. Note that we classify only a few related metrics, for illustration.

Scalability metric by Jogalekar and Woodside [11] Jogalekar and Woodside provide a scalability metric that can directly be used to answer $Q1_{scale}$ ("Does the system fulfill its requirements when the workload increases (from workload WL_X to WL_Y)?"). They use workload as the main input context factor to their metric, thus, complying to our scalability question. As an output, they determine a productivity factor that states whether system requirements can be fulfilled (using a threshold for this factor). Also this idea complies to our question. However, to calculate the productivity factor, they require knowledge about the operation costs for a given workload. In contrast, we do not limit our scalability metrics to this knowledge about costs and require it for elasticity metrics only.

Elasticity and efficiency metrics by Herbst et al. [9] Herbst et al. provide metrics that consider speed and precision (w.r.t. avoiding under- and overprovisioning) of scale-in and -out. Their ideas on speed can be used to answer our elasticity question $Q4_{elast}$ ("From a point the system violates its requirements, how long does it take before the system recovers to a state in which its requirements are met again?"). However, to detect requirement violations, they assume to have already an implemented system at hand; design-time (e.g., simulation-based) approaches do not work with their metric. Their ideas on precision fit to our elasticity question $Q5_{eff}$ ("How close to the actual resource demand can the system align the resource provisioning?"). Again, their metric requires an implemented system to determine the actual resource usage.

7. CONCLUSION

In this short research paper, we argue for the need of novel metrics for quality properties of cloud computing systems. Using the GQM method, we systematically derive an initial set of six metrics for scalability, elasticity, and efficiency. Moreover, by using our GQM plan, we classify existing metrics to make their limitations explicit.

Our systematically derived metrics help software architects, requirements engineers, testers, etc. to design and analyze cloud computing systems. Our GQM plan helps them to consider limitations of such metrics during these tasks.

Future work is directed towards the development of analysis methods and tools that enable software architects to verify the fulfillment of scalability, elasticity, and efficiency requirements of their cloud computing applications already at design time. Understanding limitations of metrics is essential for this purpose.

8. ACKNOWLEDGMENTS

This work is supported by the German Research Foundation (DFG) within the Collaborative Research Centre "On-The-Fly Computing" (SFB 901). The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant no 317704 (CloudScale).

9. REFERENCES

[1] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia. A view of cloud computing. *Commun. ACM*, 53(4):50–58, Apr. 2010.

[2] M. Becker, S. Becker, and J. Meyer. SimuLizar: Design-time modelling and performance analysis of self-adaptive systems. In *Proceedings of Software Engineering 2013 (SE2013), Aachen*, 2013.

[3] A. Berl, E. Gelenbe, M. Di Girolamo, G. Giuliani, H. De Meer, M. Q. Dang, and K. Pentikousis. Energy-efficient cloud computing. *The Computer Journal*, 53(7):1045–1051, 2010.

[4] G. Bolch, S. Greiner, K. S. Trivedi, and H. de Meer. *Queueing Networks and Markov Chains: Modeling and Performance Evaluation With Computer Science Applications*. 1998.

[5] A. B. Bondi. Characteristics of scalability and their impact on performance. In *WOSP '00*, pages 195–203, New York, NY, USA, 2000. ACM.

[6] L. Duboc, E. Letier, and D. S. Rosenblum. Systematic elaboration of scalability requirements through goal-obstacle analysis. *IEEE Transactions on Software Engineering*, 39(1):119–140, Jan. 2013.

[7] T. Erl, Z. Mahmood, and R. Puttini. *Cloud Computing: Concepts, Technology & Architecture*. Prentice Hall, 2013.

[8] E. Folkerts, A. Alexandrov, K. Sachs, A. Iosup, V. Markl, and C. Tosun. Benchmarking in the cloud: What it should, can, and cannot be. In *TPCTC*, pages 173–188, 2012.

[9] N. R. Herbst, S. Kounev, and R. Reussner. Elasticity: What it is, and What it is Not. In *ICAC '13*, 2013.

[10] S. Islam, K. Lee, A. Fekete, and A. Liu. How a consumer can measure elasticity for cloud platforms. In *Proceedings of the 3rd ACM/SPEC International Conference on Performance Engineering*, ICPE '12, pages 85–96, New York, NY, USA, 2012. ACM.

[11] P. Jogalekar and M. Woodside. Evaluating the scalability of distributed systems. *IEEE Trans. Parallel Distrib. Syst.*, 11(6):589–603, June 2000.

[12] E. Roloff, M. Diener, A. Carissimi, and P. Navaux. High performance computing in the cloud: Deployment, performance and cost efficiency. In *CloudCom '12*, pages 371–378, 2012.

[13] J. Siegel and J. Perdue. Cloud services measures for global use: The service measurement index (smi). In *SRII Global Conference (SRII), 2012 Annual*, pages 411–415, July 2012.

[14] C. U. Smith. *Performance engineering of software systems*. Software Engineering Institute series in software engineering. Addison-Wesley, 1990.

[15] R. van Solingen, V. Basili, G. Caldiera, and H. D. Rombach. *Goal Question Metric (GQM) Approach*. John Wiley & Sons, Inc., 2002.

THE CONTINUOUS AUDIT OF ONLINE SYSTEMS

Miklos A. Vasarhelyi
AT&T Bell Laboratories, 600 Mountain Ave., Murray Hill, N.J. 07974 and Rutgers University, Newark, N.J.
Fern B. Halper

AT&T Bell Laboratories, 600 Mountain Ave., Murray Hill, N.J. 07974.

Submitted to Auditing: A Journal of Practice and Theory
August 1989
Revised June 1990

The authors wish to thank the two anonymous reviewers for their constructive comments and the editor for his review of the manuscript. We would also like to thank the participants of research seminars at Columbia University, Rutgers University, the University of Kansas, the University of Nebraska, and Boston University and the attendees of the EDPAA, IIA, and AICPA professional meetings for their comments and suggestions. We are particularly indebted to Sam Parker, Chris Calabrese, Tsyh-Wen Pao, John Snively, Andrew Sherman, and Kazuo Ezawa for their work on the prototype system.

ABSTRACT

The evolution of MIS technology has affected traditional auditing and created a new set of audit issues. This paper focuses on the Continuous Process Auditing System (CPAS) developed at AT&T Bell Laboratories for the Internal Audit organization. The system is an implementation of a Continuous Process Audit Methodology (CPAM) and is designed to deal with the problems of auditing large paperless database systems. The paper discusses why the methodology is important and contrasts it with the traditional audit model. An implementation of the continuous process audit methodologyis discussed. CPAS is designed to measure and monitor large systems, drawing key metrics and analytics into a workstation environment. The data are displayed in an interactive mode, providing auditors with a work platform to examine extracted data and prepare auditing reports. CPAS monitors key operational analytics, compares these with standards, and calls the auditor's attention to any problems. Ultimately, this technology will utilize system probes that will monitor the auditee system and intervene when needed.

INTRODUCTION

This paper develops the concept and explores key issues in an alternate audit approach called the Continuous Process Audit Methodology (CPAM). The paper focuses on an implementation of this methodology, the Continuous Process Audit System, developed at AT&T Bell Laboratories for the AT&T Internal Audit Organization.

The paper is divided into four sections. In the remainder of the Introduction, changes in Management Information Systems (MIS) that affect traditional auditing are discussed. In the second section, CPAM and CPAS are described and contrasted with the traditional audit approach. The audit implications related to the introduction of a CPAS

like technology also are examined. The last section discusses some of the knowledge issues involved in the implementation of a CPAS application and suggests paths for future work.

Technology and the Auditor

Traditional auditing (both internal and external) has changed considerably in recent years, primarily as a result of changes in the data processing environment. [Roussey, 1986 ; Elliot, 1986; Vasarhelyi and Lin, 1988; Bailey et al., 1989]. These changes have created major challenges in performing the auditing and attestation function. These changes and the technical obstacles created for auditors as a result of these changes are summarized in Table 1.

TABLE 1

The Evolution of Auditing from a Data Processing Perspective

Phase	Period	Data Processing of Functions	Applications	Audit Problem
1	1945-55	Input (I) Output (O) Processing (P)	Scientific & Military applications	Data transcription Repetitive processing
2	1955-65	I, O, P Storage (S)	Magnetic tapes Natural applications	Data not visually readable Data that may be changed without traces
3	1965-75	I, O, P, S Communication (C)	Time-sharing systems Disk storage Expanded Operations support	Access to data without physical access
4	1975-85	I, O, P, S, C Databases (D)	Integrated databases Decision Support Systems (decision aides) Across-area applications	Different physical and logical data layouts New complexity layer (DBMS)

				Decisions impounded into software
5	1986-91	I, O, P, S, C, D Workstations (W)	Networks Decision support systems (non-expert) Mass optical storage	Data distributed among sites Large quantities of data Distributed processing entities Paperless data sources Interconnected systems
6	1991-on	I, O, P, S, C, D, W Decisions (De)	Decision support systems (expert)	Stochastic decisions impounded into MIS

For example, the introduction of technology precluded auditors from directly reading data from its source (magnetic tape) and, unlike paper and indelible ink, this source could be modified without leaving a trace. (phase 1 and 2 in Table 1) the advent of time sharing and data communications have allowed continuous access to data from many locations (phase 3) creating access exposures; database systems have added more complexity to auditing due to the lack of obvious mapping between the physical and logical organization of data (phase 4).

Auditors dealt with these changes by (1) tailoring computer programs to do traditional audit functions such as footing, cross-tabulations and confirmations, (2) developing generalized audit software to access information on data files, (3) requiring many security steps to limit logical access in multi-location data processing environments and (4) developing specialized audit computers and/or front-end software to face the challenge of database oriented systems.

However, MIS continue to advance in design and technology. Corporate MIS, and particularly financial systems, are evolving towards decentralization , distribution, online posting, continuous (or at least daily) closing of the books, and paperlessness [Vasarhelyi and Yang, 1988]. These changes are causing additional challenges for auditors and

TABLE 2

Database Systems and their Audit

System Characteristic	Audit (level 1)	Audit (level 2)
Database	Documentation	Data dictionary query
Database size	User query	Auditor query
Transaction flows	Examine levels	Capture sample transactions
Duplicates	Sorting and listing	Logical analysis and indexes
Field analysis	Paper oriented	Software based
Security issues	Physical	Access hierarchies
Restart & Recovery	Plan analysis	Direct access
Database interfaces	Reconciliation	Reconciliation and transaction follow-through

Audit work on these systems is constrained by strong dependence on client system staff (for the extraction of data from databases) and typically entails reviewing the manual processes around the large application system. In traditional system audits these procedures were labeled as “audit around the computer”. These procedures, are labeled as “level 1” in Table 2 and are characterized by examination of documentation, requests for user query of the database, examination of application summary data, sorting and listing of records by the user (not the auditor), a strong emphasis on paper, physical evaluation of security issues, plan analysis for the evaluation of restart & recovery and manual reconciliation of data to evaluate application interfaces. Level 2 tasks, described in Table 2, would use the computer to perform database audits as well as eliminate the intermediation by the user or systems people (auditees) in the audit of database systems. This hands-on approach utilizes queries to the data dictionary, direct use of the system by the auditor and would rely on transaction evidence gathered by the auditor using the same database technology. The level 2 approach reduces the risk of fraudulent (selective) data extraction by the auditee and allows the audit to be conducted more efficiently if the auditor is well versed in database management. Furthermore, audit effectiveness is increased because the auditor has greater flexibility in the search for evidence and it is not obvious to the auditee what data are being queried by the auditor (resulting in improved deterrence of fraud). Differences in desired audit approach and the technological tooling necessary for performing level 2 tasks led to the development of some of the concepts used for Continuous Process Auditing.

CONTINUOUS PROCESS AUDITING

There are some key problems in auditing large database systems that traditional

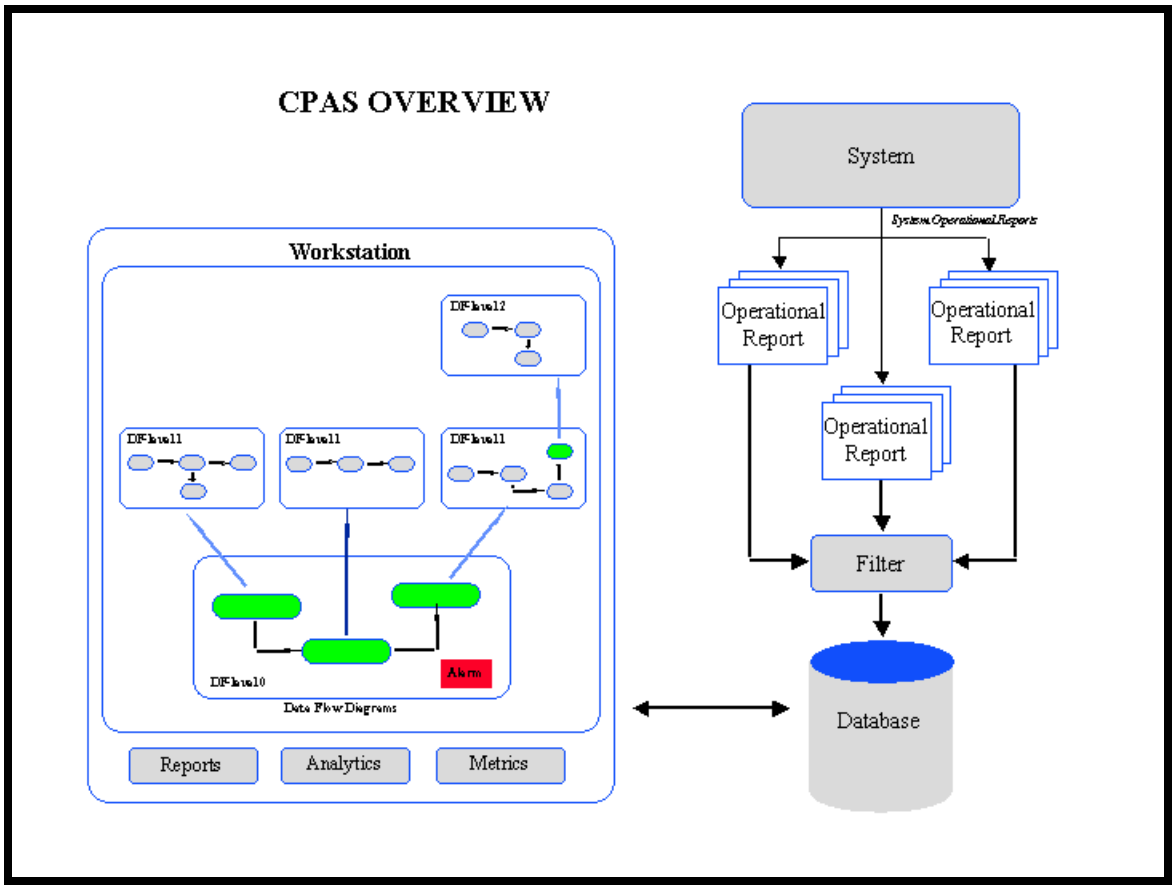
may imply an obtrusive intrusion on the system and can result in performance deterioration. The installation of these monitoring devices must be planned to coincide with natural life-cycle changes of major software systems. Some interim measures should be implemented to prepare for online monitoring. The approach adopted at AT&T, with the current CPAS prototype, consists of a data provisioning system and an advanced decision support system.

Data provisioning can be accomplished by three different, though not necessarily mutually exclusive methods: (1) data extraction from "standard" existing application, reports, using pattern matching techniques; (2) data extraction form the file that feeds the application report; and (3) recording of direct monitoring data. The approach used in CPAS entails first a measurement phase where intrusion is necessary but the audit capability is substantially expanded.

Measurement. Copies of key management reports are issued and transported through a data network to an independent audit workstation at a central location. These reports are stored in raw form and data are extracted from these reports and placed in a database. The fields in the database map with a symbolic algebraic representation of the system that is used to define the analysis. The database is tied to a workstation and analysis is performed at the workstation using the information obtained from the database. The basic elemtns of this analysis process are described later in the paper.

Monitoring. In the monitoring phase, audit modules will be impounded into the auditee system. This will allow the auditor to continuously monitor the system and provide sufficient control and monitoring points for management retracing of transactions. In current systems, individual transactions are aggregated into account balances and complemented by successive allocations of overhead. These processes create difficulties in balancing and tracing transactions.

The AT&T CPAS prototype uses the "measurement" strategy of data procurement. This is illustrated in Figure 1. The auditor logs into CPAS and selects the system to be audited. The front end of CPAS allows the auditor to look at copies of actual reports used as the source of data for the analysis. From here the auditor can move into the actual analysis portion of CPAS. In CPAS, the system being audited is represented as flowcharts on the workstation monitor. A high level view of the system (labeled DF level 0 in Figure 1) is linked hierarchically to other flowcharts representing more detail about the system modules being audited. This tree oriented view-of-the-world which allows the user to drill down into the details of a graphical representation is conceptually similar to the Hypertext approach [Gessner, 1990]. The analysis is structured along these flowcharts leading the auditor to think hierarchically.



Analysis. The auditor's work is broken down into two phases: first, the startup stage where he/she works with developers, users, and others to create a view of the system, abd second, the use stage when he/she actually uses the system for actual operational audit purposes. The auditor's (internal or external) role in this context is not very different from its traditional function.

At the setup stage, the auditor acts as an internal control identifier, representer, and evaluator using existing documentation and human knowledge to create the system screens (similar to flowcharts) and to provide feedback to the designers/management. Here, audit tests, such as files to be footed and extended or reconciliations to be performed, as well as processes to be verified, are identified. Unlike the traditional audit process, the CPAS approach here requires the "soft-coding" of these processes for continuous repitition. Furthermore, at this state, the CPAS database is designed, unlike in the traiditonal process, standards are specified and alarm conditions designed.

In the use stage, the system is monitored for alarm conditions and the alarm conditions are investigated when they arise and the symptoms and diagnostics identified and impounded into the CPAS knowledge base. The current baseline version of CPAS provides auditors with some alarms for imbalance conditions, the ability to record and display time-series data on key variables, and a cries of graphs that present event de-composition.

This logical view of the system can be associated with diagnostic analytics that count the number of exceptions and/or alarms current in the system. Detailed information about

each main module is available at lower levels through a drill-down procedure. This information is presented primarily as metrics. analytic, and alarms.

Metrics

Metrics are direct measurements of the system, drawn from reports, in the measurement stage. These metrics are compared against system standards. If a standard is exceeded, an alarm appears on the screen. For example, in the auditing of a billing system, the number of bills to be invoiced is extracted from a user report. The number of bills not issued due to a high severity error, detected by the normal data processing edits, is captured as well as the total dollar amount of bills issued. These three numbers are metrics that relate to the overall billing process.

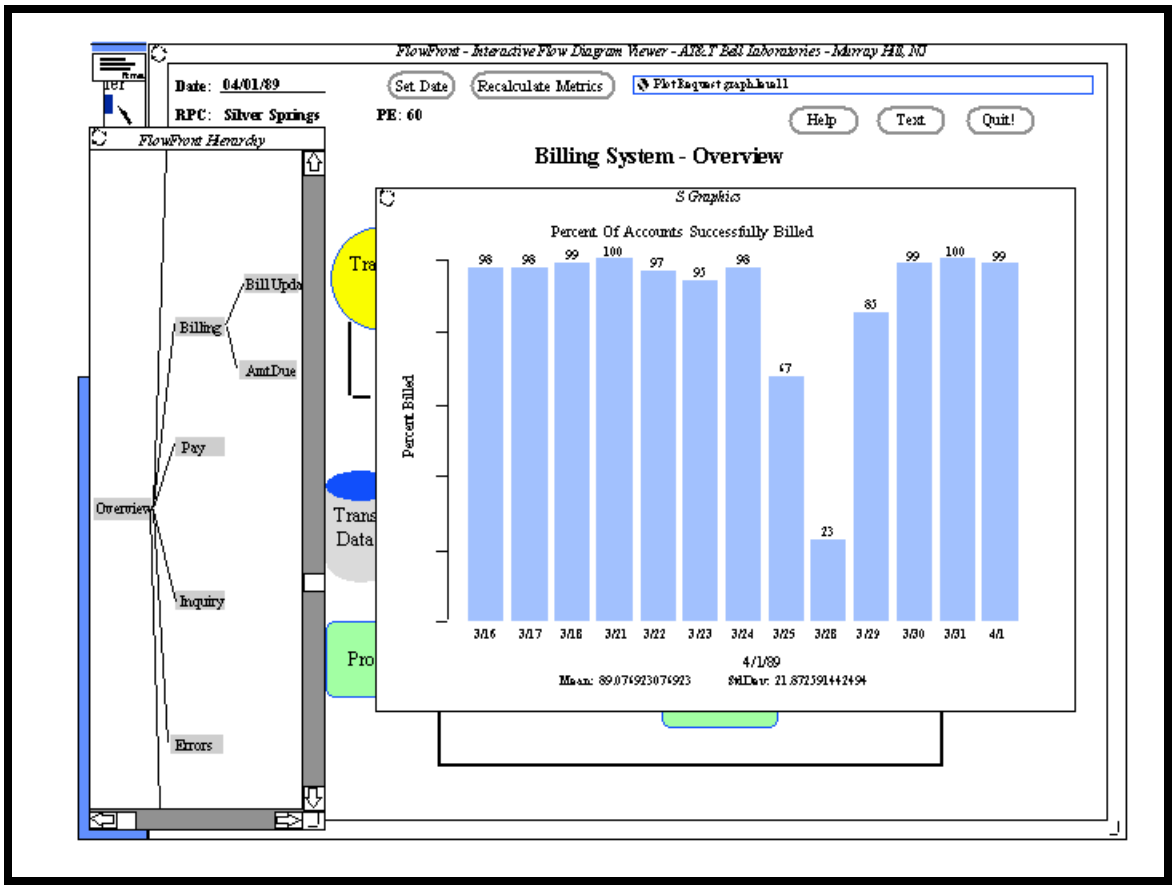
Analytics

Analytics are defined as functional (natural flow), logical (key interaction), and empirical (e.g. it has been observed that ....) relationships among metrics. Specific analytics, related to a particular system module can be derived from the auditor, management, user experience, or historical data from the system. Each analytic may have a minimum of three dimensions: 1) its algebraic structure, 2) the relationships and contingencies that determine its numeric value at different times and situations and 3) rules-of-thumb or optimal rules on the magnitude and nature of variance that may be deemed as “real variance” to the extreme of alarms. For example, a billing analytic would state that dollars billed should be equal to invoices received, minus values of failed edits plus (or minus) the change of the number of dollars in retained invoices. The threshold number of expected invoices for that particular day or week (allowing for seasonality) must be established to determine whether an alarm should be fired.

Alarms. An alarm is an attention-directing action triggered, for example, when the value of a metric exceeds a standard. Actual experience with these issues indicates that *several* levels of alarms are desirable; (I) minor (type I) alarms dealing with the functioning of the auditing system; (2) low-level operational (type 2) alarms to call exceptions to the attention of operating management (3) higher-level (type 3) alarms to call exceptions to the attention of the auditor and trigger 'exception audits:' and (4) high-level hype 4) alarms to -am auditors and top management of serious crisis.

For example, a type I alarm may be triggered if two sets of data are produced by the audited system, for the same module, for the same day, and it is unclear from information given which data to load into the database. Of course, cycle and re-run information should be clearly passed along with the data but sometimes this will not be as clean as expected. A type I alarm might also be triggered if the reports change format and data extraction procedures need to be modified. These Type I alarms will need to be acted upon immediately, usually with a call to the system administrator or system management organization.

A type 2 alarm might be triggered if data pertaining to the same process are inconsistent. For example data from many different reports might he used to perform an intramodule reconciliation. The data must come from different jobs in order for the



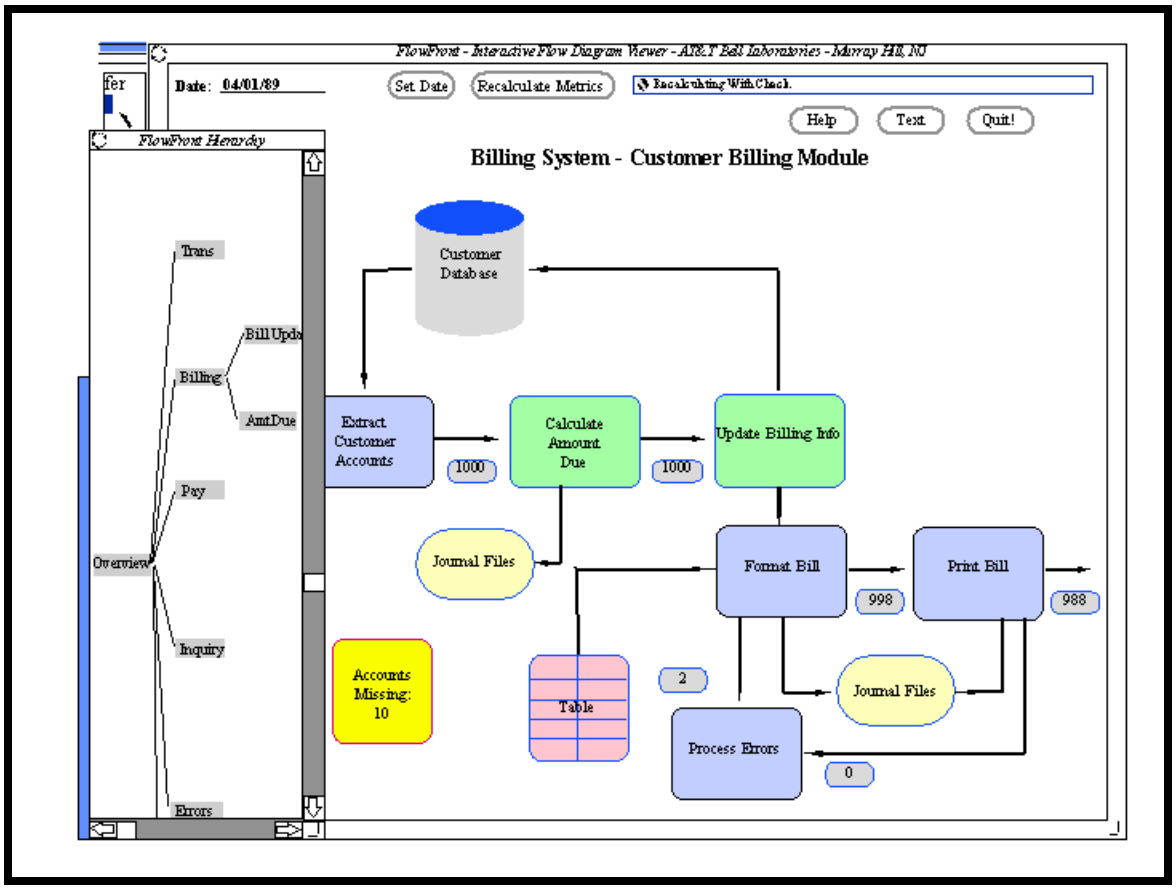
The CPAS application may be testing that the following controls are in place: (1) completeness and accuracy of input; (2) completeness and accuracy of update; (3) timeliness" of data arriving to die system and h timeliness of system processing; (4) maintenance of data in the database; (5) accuracy of computer programs; and (6) reasonableness of the data. For example, the auditor might have defined tests (and had them built into the CPAS application) to answer the following questions:

Were all transactions sent to the biller, received? Can all of the transactions be accounted for? Were all of the trans actions loaded into the Process Transactions module? Were they loaded correctly?

How many transactions were in error? Has the error threshold been exceeded? How long does it take errored transactions to re-enter the system?

Were all transactions posted to the database correctly? Were all the trans. actions initiated, executed, and recorded only once? Can all of the transactions that entered the system be accounted *(or (i.e.*, either on the database or in an *error* file, or rejected back to the source)? How accurate are the data that were loaded to the database (i.e., does the sum of! he dollars on the database match what was to be posted to the database)? Are all databases synchronized?

Were: the bills calculated properly? How reasonable are the amounts billed? Were all customers who *were* supposed to be billed actually billed?



The audit" may wish to look at the history of the reconciliation. Figure 5 is a two level time-series showing the number of accounts lost and the total number of accounts billed for a dive-week period ending 4/l/90.The graph indicates that the out-of-balance condition occurred once at the beginning of the period and again on 4/1/90. M condition appeared to have bun corrected at the beginning of the period, since the reconciliation did not fail again until the current day's processing. The auditor should reset the date to 3/11/90 and check the metrics to determine if the reconciliation failed for the same reason that it did on 4/1/90. This could indicate inadequacy of controls or poor compliance with internal controls. More detailed analytics and we metrics relating to the actual billing process and the interface between this module and other modules in the system are found at different levels. This information, taken together, presents an integrated diagnostic view of the system being audited."

Insert Figure 3

Complementing the actual hands-on audit work is an auditor platform, accessible at any level, which can include a series of different functions. This platform should ultimately contain at least a statistical package, a graphics package, a spreadsheet package (including a filler to the database), a report generator, and a teat editor. These tools can be used for *ad hoc* analysis or be linked to the “wired-in” procedures in CPAS. An even other technological environment may incorporate

CPAM requires long-term monitoring and reaction to emerging evidence, something that, with limited experience, is difficult to manage. Given this, the issue of resistance to change may arise. This can be handled by the issuance of an audit manual that describes how to audit with CPAS and extensive training and technical support for the auditors.

Ideally, management also its version of CPAS, so they are aware when major problems occur in their system. Auditors could browse their own CPAS (with independent analytics) on a periodic basis and follow up on any alarm conditions to see what management has done about them.

Future work on CPAS will focus on increasing the quality of auditor work by integrating the auditor platform with the auditor workstation, increasing the use of monitoring probes, improving the quality of the auditor heuristics, and impounding more expertise into the system.

The introduction of real-time systems require that the auditor be able to attest to the system of internal accounting controls at different points in time. Continuous process auditing can effectively help the auditor to evaluate these controls, but will require substantive changes in the nature of evidence, the types of procedures, the timing, and the allocation of effort in audit

REFERENCES

Alter, S. , Decision Support Systems: Current Practice and Continuing Challenges (Addison-Wesley Publishing Company, 1980).

Bailey, A. D., G. L. Duke, G. E. Gerlach, C. E. Ko, R. D. Meservy and A. B. Whinston, "TICOM and
the Analysis of Internal Controls," The Accounting Review,
(April 1985), pp. 186-201.

---and R. D. Meservy, P. E. Johnson, "Internal Control Evaluation:
A Computational Model of the Review Process," Auditing: A Journal of Practice and Theory, (Autumn 1986), pp. 44-74.

---and K. Hackenbrack, P. De and J. Dillard, "Artificial
Intelligence, Cognitive Science and Computational Modeling in
Auditing Research: A Research Approach," Journal of Information
Systems1 (Spring 1987).

---and L. E. Graham, and J. V. Hansen, "Technological
Development and EDP" in
Abdel-Khalik, A. R. & Solomon, I. "Research Opportunities in Auditing:

The Acceptance and Adoption of Continuous Auditing by Internal Auditors: A Micro Analysis

Miklos A. Vasarhelyi, **Michael Alles, Siripan Kuenkaikaew**, Rutgers Business School

James Littley, KPMG

Working Paper

Abstract:

The umbrella of "advanced technology" covers a range of techniques widely used in the U.S. to provide strategic advantage in a very competitive business environment. There is an enormous amount of information contained within current-generation information systems, some of which is even processed on a real-time basis. More importantly, the same holds true for actual business transactions. Having accurate and reliable information is vital and advantageous to businesses, especially in the wake of the recent recession. Therefore, the need for ongoing, timely assurance of information utilizing continuous auditing and continuous control monitoring methodologies is becoming more apparent. To that end, we have conducted interviews with 22 internal audit managers and 16 internal audit staff members at 9 leading internal audit organizations to examine the status of technology adoption, to evaluate the development of continuous auditing, and to assess the use of continuous control monitoring. We found that several companies in our study were already involved in some form of continuous auditing or control monitoring while others are attempting to adopt more advanced audit technologies. We also made a large number of surprising observations on managerial, technology training and absorption, and other issues. Within the According to the audit maturity model (Vasarhelyi et al, 2009), all of the companies were classified between the "traditional audit" stage and the "emerging stage", not having yet reached the "continuous audit" stage. This paper, to our knowledge, is the first to study CA technology adoption in a micro level by an interview approach.

1. Introduction

> *"Continuous auditing is a progressive shift in audit practices towards the maximum possible degree of audit automation as a way of taking advantage of the technological basis of the modern firm in order to reduce audit costs and increase audit automation. Given the emphasis on the transformation of the entire system of auditing, the development of CA requires a fundamental rethink of all aspects of auditing, from the way in which data is made available to the auditor, to the kinds of tests the auditor conducts, how alarms are dealt with, what kinds of reports are issued, how often and to whom and many other issues, the importance of some of which will only become apparent as CA is implemented. It is important for the profession and other stakeholders to start thinking about the impact of CA on auditing now, when it is easier to put in place the foundations for this change rather than when technologies and practices have already become established."*
> Vasarhelyi et al (2010)

Continuous auditing (CA) is a methodology that enables independent auditors to provide written assurance on a subject matter using a series of auditors' reports issued simultaneously with, or a short period of time after, the occurrence of events underlying the subject matter (CICA/AICPA, 1999). In order to achieve its goal of reducing the latency between the occurrence of the business transaction and the provision of assurance on that transaction, continuous auditing relies heavily on such information technologies as ERP systems, data analysis and business intelligence softward, web application server technology, web scripting solutions and ubiquitous database management systems with standard connectivity (Sarva 2006).

Alles et al (2008) indicate that CA has moved over the last two decades from an academic vision (Vasarhelyi and Halper, 1991; Groomer and Murthy, 1989) to a vibrant and growing area of audit practice. Given the technological basis of CA, perhaps the best metric of the increasing acceptance of continuous auditing as an audit methodology is the nearly 100,00 hits that the term generates on Google (as of July 1, 2010)—more than double the number of hits returned just a year earlier. Practitioners and software vendors (such as SAP, ACL, Caseware, Approva and Oversight Systems) now outnumber academic researchers as attendees at the biannual global CA conferences organized by Rutgers University in the USA. Brown et al (2007) undertake a survey of more than sixty research papers into various aspects of continuous auditing.

The most striking confirmation of the importance and impact of CA on audit practice came from a set of surveys in 2006. A PricewaterhouseCoopers survey from June of that year finds that: *"Eighty-one percent of 392 companies responding to questions about continuous*

auditing reported that they either had a continuous auditing or monitoring process in place or were planning to develop one. From 2005 to 2006, the percentage of survey respondents saying they have some form of continuous auditing or monitoring process within their internal audit functions increased from 35% to 50%—a significant gain."[1]

A similar survey jointly undertaken by ACL and the Institute of Internal Auditors also shows that interest in CA is increasing rapidly, with 36% of responding firms stating that they have adopted a continuous auditing approach across all of their business processes or within select areas, and with another 39% planning to do so in the near future. The latter survey concludes: "*Whatever the reasons organizations may have had for neglecting continuous auditing in the past, regulatory demands, the push for real time financial reporting and the drive to automate resource draining manual audits are nudging them to adopt it now."*[2]

While these surveys provide evidence on the growing acceptance of continuous auditing, the macro-level nature of the surveys does not allow a full understanding of how precisely the survey subjects are implementing CA and the challenges and opportunities that these firms face when doing so. Moreover, continuous auditing is a concept rather than a well defined technological tool or practice and hence it is not clear what the responding firms actually mean when they say that they "*had a continuous auditing or monitoring process in place"* or the extent to which they have *"adopted a continuous auditing approach across all of their business processes"*.

The purpose of this paper is to undertake a micro-level study of the state of adoption and implementation of continuous auditing systems by internal auditors. Such an in-depth examination of how auditors actually perceive and use an emerging technology is meant to provide guidance to both audit practitioners and researchers about how CA is evolving as what was once a purely academic concept meets the reality of business. Despite the greater insights that they potentially provide, micro-level studies are less common that macro-level surveys because of the inherent difficulty in getting access to users and obtaining their cooperation. Thus the contribution we make is the unique degree of access we were able to obtain with internal auditors at some of the leading corporate users of continuous auditing systems.

---

[1] Available at: CFO.com, June 26, 2006.
[2] Business Finance Magazine, August 1, 2006. Available at: http://businessfinancemag.com/article/upfront-continuous-auditing-ready-prime-time-0801

This paper is one output of a major research program undertaken between the Continuous Auditing and Research Laboratory (CARLAB) at Rutgers Business School and its sponsor and partner, KPMG, whose aim was to obtain a 360-degree view of the state of the art of CA, encompassing its use by both external and internal auditors. This particular paper focuses on how CA is being implemented by internal auditors at firms known to be pioneers in its use. Thanks to the involvement of KPMG, as well as the Rutgers CARLAB's own contacts, the research team had privileged access to senior members of the internal audit departments of nine leading global businesses. We interviewed 22 internal audit managers and 16 internal audit staff from these companies using a detailed interview guide to obtain in depth and comparable information about how these internal auditors perceive the usefulness of CA, its ease of use and its costs and benefits.[3] From our interview results, most CA implementations remain in the preliminary phrases. By identifying the drivers and barriers that affect the adoption of continuous auditing and continuous control monitoring in organizations, we hope we provide a better understanding of the stage of development and usage of the methodology.

Section 2 explains the methodology of the field study. Section 3 provides a discussion of relevant research literatures, which leads the construction of the interview guide and the audit maturity model that is used to frame the discussion of the results of the interviews. Section 4 discusses our results, while section 5 offers concluding comments.

2. Motivation and Methodology

Our objective in this paper is to understand the current state of the art in continuous auditing adoption by internal auditors. As discussed in the Introduction, much of the information on this subject has come up to now from non-academic large scale surveys conducted by external audit firms or software vendors. Their methodology and the impact, if any, of their vested interests are hard to discern in their published reports.

It is even harder to assess what their conclusions imply. From our own experience working in very large companies in which a small scale CA project is undertaken by one group of internal auditors in one part of the business, it is not at all clear what response would be given by someone asked to fill out a CA survey in another part of the company, especially if that

[3] It should be emphasized that KPMG is not the external auditor at any one of our subject businesses.

was truly a guide and not a constraint, and served as a floor on the scope and coverage of the conversations with the internal auditors and not a ceiling. The guide also served as a medium for crystallizing the research objectives of the study by having its content shaped by a wide ranging survey of the literature on technology adoption and acceptance.

The sample of firms used in the study was chosen by the researchers in collaboration with KPMG on the basis of two criteria: that they were anecdotally considered to be experienced users of CA systems and that they spanned a broad range of businesses. Thus the sample consisted of an insurance company, two banks, two technology firms and four consumer goods companies. In addition, one of the banks is entirely based outside the USA. These firms are obviously not randomly chosen, but the aim here is not to analyze firms that use CA versus those that don't, but rather to study in depth the experiences and motivations of the firms that have adopted CA.

3. Technology Acceptance and Adoption

3.1 The TAM Model and the Construction of the Master Interview Guide

Our objectives in this paper are to understand the factors that shaped the acceptance of continuous auditing as a new methodology in internal auditing, and to provide some perspective for our results through developing a metric on the degree of adoption of the technology. Achieving these two goals requires us to draw on several strands of the research literature that examine the reaction of users to new technologies.

The most widely used paradigm for studying the acceptance of technology is the Technology Acceptance Model (TAM), first proposed by Davis (1985, 1989), and which is the subject of over 700 subsequent published research papers (Bagozzi, 2007). The TAM model assumes that acceptance is driven by the user's attitude towards the technology and that attitude is a function of its perceived usefulness (PU) of the technology and the perceived ease of use (PEU) of the technology. Davis (1989) defines PU as *"the degree to which a person believes that using a particular system would enhance his or her job performance",* while PEU is defined as *"the degree to which a person believes that using a particular system would be free from effort"*.

While there have been numerous variations of this basic model, PEU and PU remain the essential drivers of the TAM. Davis's (1985, 1989) model was intended to be applied, however,

and not remain a theoretical construct, and the hundreds of papers that have followed have used the TAM framework to examine numerous technologies, including email programs, voice mail, operating systems and office productivity software (Lee et al, 2003; Legris et al, 2003). Some of the criticisms leveled at this research are that it relies too heavily on students as subjects and that it depends largely on self-reported behavior, which may not correspond to actual usage of the technology (Chuttur, 2009).

While the TAM model is clearly relevant to our examination of continuous auditing, we cannot apply it in the standard way that it is has been in the TAM literature. Most applied TAM papers are essentially surveys of hundreds of users of a very specific technology, with numerous overlapping questions used to obtain measures of PU and PEU. Thus, Davis (1989) examined the attitude of 112 IBM employees to an email program by asking them 10 questions related to PU and another 10 on PEU. His questions on PU included *"Using electronic mail improves the quality of the work that I do"* and *"Overall, I find the electronic mail system useful in my job",* while the PEU questions ranged from *"I find it cumbersome to use the electronic mail system"* to *"Overall, I find the electronic mail system easy to use"*.[5]

Given that the strength of our approach to studying CA acceptance is the opportunity to talk in depth with CA users, the use of such a mechanistic set of questions to assess PU and PEU would have been counterproductive. Indeed, that approach would have been better suited for the kinds of surveys of CA use that we are trying complement rather than replicate. Moreover, a fundamental difference between CA and the kinds of technologies that have been examined in the TAM literature is that CA is an emerging business practice and not a particular piece of software. While there are indeed CA-related products, such as ACL or Approva, our objective is not to study their implementation in particular, but rather, how internal auditors perceive the set of technologies and audit practices that comprise continuous auditing, and which together differentiates CA from the standard way in which internal auditing has been carried out. Not having a specific product makes it much harder to ask the detailed, overlapping questions of the sort that the TAM literature relies on for obtaining data and conducting its statistical analyses.

[5] Looking back at Davis (1989), after 20 years, makes one realize that the "acceptance" that he is examining has to be put into context and is entirely relative to its time and place. Who, today, would bother to ask such questions about something that is so taken for granted as email? There is a lesson there, perhaps, for CA too if one looks forward sufficiently far ahead.

As Alles et al (2006, 2010) and Teeter and Brennan (2010) indicate, early adopters of CA will likely simply automate existing audit practices, going after the "low hanging fruit" of processes that are easily and simply automatable. Once the benefits of that are apparent, CA will grow into further areas of the audit, but this will necessitate reengineering the audit to make more processes capable of being automated. Finally, by the mature stage, much of the internal audit will be automated and conducted independent of location, with the human auditors focusing only on the audit tasks that demand the most subjectivity, such as in assessing the "tone at the top".

Each of these four stages can be further classified along the following seven categories:

1. **Audit Objective:** The scope of audit tasks that is undertaken by CA systems.
2. **Audit Approach:** The extent to which audit outputs shifts from being periodic to being undertaken continuously.
3. **Data access:** Level of access of the internal auditors to the firm's data systems.
4. **Audit automation**: The degree to which audit processes are automated.
5. **Audit and management overlap:** The extent to which internal auditors rely on IT systems intended for management use.
6. **Management of audit function:** Organizational relationship between the IT internal audit, the finance audit and other compliance departments.
7. **Analytic methods:** Degree of technical sophistication of analytical procedures that internal audit performs.

4. Results and analysis

Perceived usefulness (PU)

The companies in the study have different level of CA/CM adoption and various perspectives of the perceived usefulness of CA/CM. Automated and integrated technologies aid internal auditors in several ways and enable greater audit efficiency. The audit-aid technology implementation is initiated and supported by the head of the internal audit department or upper level management. The internal audit department of each company is responsible for monitor and assesses internal control effectiveness and report the assessment result in exception reports. If the irregularity event has been captured, CA/CM systems will generate alarm which will notify internal auditors and management.

One of the companies has implemented a system setting monitoring tools used by IT service staff. However, it is not CA/CM system yet. The internal audit director informed that "*Our IT service colleague already has the tools that monitor the configurable settings for the systems, databases, and network. What we need to do is work with them to get them into where they are continuously monitoring. Then, our audit can focus on how we are going to deal with the exceptions.*"

All of the companies in our study have to comply with SOX requirement, and they have specific divisions to monitor and ensure the compliance. Even though SOX requirement is not the main reason for the companies to implement CA/CM, they found that it tremendously facilitates SOX requirement. SOX review is very detailed, complicated and time consuming tasks. Interviewees reported that CA/CM assists the review activities and reduces the time allocated to SOX compliance. For example, the audit department of one company developed a monitoring tool to review the ERP system for both general internal control purposes and SOX compliance. This tool helps internal auditors work efficiently and supports comparison and benchmarking of the control components. External auditors are therefore able to rely on the work of internal auditors, which reduce time and effort required from both parties. As management mentioned *"…we developed out the tools that can dump everything out on the table... so much of our objective for this has been SOX and driven by [external auditor]."*

CA maturity model

By the metrics of the audit maturity model, the current level of adoption of CA by the internal audit departments of the companies in the sample is between stages 1 and 2, as shown in Figure 2. In other words, quite different to the findings of the PwC and ACL/IIA surveys, CA implementation is still largely at the introductory stage, with most users being best described as early adopters.

Looking in depth at our categories of CA maturity model, we can analyze the degree of CA/CM adoption as follow.

1. Audit Objectives

In addition to providing reasonableness assurance on the financial statements and financial reports, the internal audit department wants to implement and enforce effective control monitoring to the companies, and this objective is progressing. If CA technology is fully adopted, internal auditors will have continuous auditing and continuous control monitoring in place and do audit by exception. The audit objective of all companies is to have effective control monitoring and auditors try to assurance the quality of the controls.

2. Audit Approach

Most of the companies are in between stage 1 and 2 of the model. They have traditional interim and year-end audit, while try to review over key risk area and monitoring on those area more frequently. However, one company reaches maturing stage 3 as it implemented continuous control monitoring and has alarms to notify irregularity in the system. The company has been implementing continuous auditing for almost 10 years in each office location by constructing audit routines in the mainframe, and monitoring iterative processes. The company monitors over 5 million customer accounts on a daily basis, and the system sends out about 6 thousand alerts a month. Internal auditors analyze the alarm and report to management.

3. Data Access

Several companies extract key data periodically to support audit cycle. Even though internal auditors have more access to data than they were in the past, with cooperation from business data owner and IT department, limitation still exists. One of the interviewed companies'

management explained that they have 25 SAP-based systems installed across the organization. Each instance is managed by a different SAP team, and data extraction is done on a monthly basis using in-house software built on top of the SAP system. Data calculation then computes via the ABAP protocol, and reports are generated. The system can keep aggregate data for at least 13 months and detail data for 3 months. The company has an enterprise data warehouse, containing financial information, but usage is limited due to reconciliation issues.

4. Audit automation

Four out of nine companies are in stage 2 of this domain. They deploy some audit management software and have electronic working paper software to manage audit documentation. In this study, Microsoft office is not considered as an electronic working paper. The audit management software allows project manager to follow audit task status and work that assigned to audit members. Audit automation also embraces automatable audit processes as in the example of Bank1 that implemented continuous control monitoring module and has the alarm system as mentioned in the audit approach section.

5. Audit and management overlap

From the interview, we found that an internal audit department of each company shares audit reports and monitoring results in a form of reporting to management. Even though management has access to some monitoring systems, monitoring task is responsible by an internal audit department or audit-like departments. As we received comments from interviewees, they mentioned about management support and buy-in; visibility among the management team; each unit reports to its head and tasks among different departments often overlap. However, one company is implementing information sharing software in order to facilitate information sharing between departments.

6. Management of the audit function

Every company sets up IT audit function to audit the systems in addition to financial reporting systems. IT audit functions of some companies are still in the initial state and have only a few resources and capability, while these functions of other companies are more advance. As companies implement Enterprise Resource Planning (ERP) system such as SAP, more controls

are automated. Thus, IT audit function has a critical role to audit the system. One company developed in-house audit system to specifically monitor the ERP system. This system also facilitates external auditor including SOX audit.

7. Analytical methods

Most of the companies emphasize on auditing financial ratio at an account level. However, some companies progress to transaction level and develop key performance indicator (KPI) and dashboard to support monitoring purpose. For instance, the KPI tool of one company utilizes both leading indicators, such as percentage of system uptime, and lagging indicators, such as a number of incident tickets. The monitoring tool has the ability to generate graphs that show trends and compare activities of selected transactions. The KPI benchmark report is generated monthly and compares operations from two periods.

In all, there is opportunity for the companies to progress toward the higher stages. They can have more automated tools to support an audit review process, concentrate in a level of analytical procedures, invest more in information technology and personnel, and improve the level of cooperation between each unit.

4. Conclusion

With the emerging of a continuous auditing and continuous monitoring methodology, an on-going, timely review of financial data and internal control of the company is enhanced. From the interviews with internal audit managers of leading organizations, we could understand and evaluate the status of technology adoption and development in this area. There are some factors that affect the adoption such as management support and employee knowledge. To perform an audit review and data analysis efficiently, an internal auditor needs a certain level of information system and data access either via application programs or via extractions by the IT department. Generally, internal auditors are responsible for monitoring the internal controls with a continuous control monitoring technology, and report any exception to the auditee's management. Thus, internal auditors need some skills and knowledge about the technology used and the audit practice. For that purpose, training is provided to support their work and enhance their ability. Continuous auditing and continuous control monitoring technology also facilitates SOX

compliance. Field work time and iterative tasks can be reduced. All of the internal audit departments have some kind of tools and audit automation to support their work, such as electronic working papers and data analysis tools. Some companies are more advanced and have a continuous monitoring tool and an alarm system.

Based on the result of the interviews, the companies can be classified according to the audit maturity model to evaluate the status of continuous auditing and continuous monitoring. Most of them are ranked between stage 1, traditional audit, and stage 2, emerging. This means that although they have certain level of CA/CM, they are just in the initiation phrase, and there is opportunity for development in the future. This result is strikingly contrasted with the PwC survey, which stated that a large number of companies had continuous auditing in place.

The limitation of this research is the generalization because of the small samples. This study can be extended in two ways. First, a structural survey research can be conducted to get more detail characteristics and behavior of technology adoption by organizations. Additional measurements can be included, and the questionnaire method will get more sample size than interview technique. Second, the follow up interview with the organizations would provide useful information about the technology adoption trend and progress.

References

Arnold, V., and S. Sutton. 1998. The Theory of Technology Dominance: Understanding the impact of intelligent decisions aids on decision makers‟ judgments. Advances in Accounting Behavioral Research 1: 175-194.

Bagozzi, R. P. 2007. The legacy of the technology acceptance model and a proposal for a paradigm shift. Journal of the Association for Information Systems, 8(4), 244-254.

Behn, B. K., D. L. Searcy, J.B. Woodroof. 2006. A Within Firm Analysis of Current and Expected Future Audit Lag Determinants. Journal of Information Systems 20 (1): 65-86.

Bohlen, Joe M.; Beal, George M. 1957, "The Diffusion Process", Special Report No. 18 (Agriculture Extension Service, Iowa State College) 1: 56–77.

Canadian Institute of Chartered Accountants. 1999. Research Report on Continuous Auditing. Toronto, Canada.