CLOUD PROVENANCE AND CORNERSTONE DATA AMALGAMATION FOR SECURITY INTELLIGENCE

ASIF IMRAN
MSSE0105

A Thesis
Submitted to the Master of Science in Software Engineering (MSSE) Program Office
of the Institute of Information Technology, University of Dhaka
in Partial Fulfillment of the
Requirements for the Degree

MASTER OF SCIENCE IN
SOFTWARE ENGINEERING

Institute of Information Technology
University of Dhaka
DHAKA, BANGLADESH

CLOUD PROVENANCE AND CORNERSTONE DATA AMALGAMATION FOR SECURITY INTELLIGENCE

ASIF IMRAN

Approved:

Signature	*Date*
Supervisor: Dr. Kazi Muheymin-Us-Sakib	
Examination Committee Head: Dr. Mohammad Shoyaib	
Committee Member: Dr. Md. Zulfikar Hafiz	
Committee Member: Dr. Muhammad Mahbub Alam	

To *Dr. Kazi Muheymin-Us-Sakib*, my research supervisor

Abstract

Open source cloud computing provides free, on-demand, rapidly scaling, ubiquitous computing and data storage services that promises a significant number of prospective customer base. However, the largely distributed nature and growing demand for open source cloud makes the infrastructure an ideal target for malicious attacks and unauthorized file transfer activities. In case of any nefarious activity, the Cloud Forensic Experts evaluate provenance information. However, the provenance is prone to tampering. Analyzing the cloud provenance only is not enough in digital forensic investigations. The provenance need to be enriched through melding those with updated and fresh security data from the web. Hence besides securing provenance, amalgamation of web cornerstone data with those for effective Cloud Security Intelligence (CSI) need to be addressed as well. This in turn will promise to accelerate the acceptability of open source cloud to the customers through increasing its security and integrity.

The proposed scheme ensures the security of provenance in a series of steps, the first of which involves binding the provenance journal data and user data through which the provenance was derived. Active-threading mechanism was proposed to secure the captured provenance. This mechanism considers each provenance journal as individual object and those are bounded with multiple user data if required. In this way the problem of ensuring the security, integrity and trust of the open source cloud has been addressed. The proposed cloud provenance securing framework is a generalized mechanism that incorporates an independent *Auditor* process to receive and secure the provenance journals from the *Provenance Owner* (*PO*). The *Auditor* process is trustworthy and verified by both the *PO* and cloud customers. Hence the integrity of provenance information can be ensured by the proposed framework.

Cloud provenance can be secured through the proposed generalized framework that is applicable to both system and application level. However, once provenance is secured it needs to be correctly analyzed during Digital Forensic Investigation (*DFI*). Effective analysis of provenance during digital investigation cannot be achieved through a generalized framework, hence it requires more specialized solution.

This brings about the second research issue addressed here that involves amalgamating provenance with unstructured web data (referred as cornerstone data in this thesis) for improved security decisions. Merging provenance with unstructured web data provides the advantages of fresh information and high validity. Secondly, the web sources selected for obtaining the unstructured cornerstone data for provenance analysis is not applicable in all cases. For this research, detection of specific real life worm attack model for nefarious malware activities in the cloud have been taken as the research case. Specific attributes like owner rating, user rating and freshness have been considered as verification criteria for the algorithms to increase the acceptance of data in the case. Next effective-descriptive set theory have been proposed for identifying valid web data that treats sources as specific set elements, each set of applications consists of a subset of processes from which the cornerstone data are obtained [1], [2].

Collection of provenance from the application layer are obtained for six real life applications provided as Software as a Service (SaaS) namely Inventory Management Software (IM), Accounting Management Software (AM), Human Resource Management Software (HRM), Office Document Management Software (ODM), Sales Management Software (SM), Customer Relationship Management Software (CRM). The proposed model uses causal chain based and taxonomized provenance amalgamation with cornerstone data obtained from the real life SaaS applications. Since the processes associated with the applications discussed above like database read, write, copy, etc are dynamic and those occur in real-time, the causal chain mechanism will have less overhead as found in the research because of its less layer of encryption [3]. The onion framework provides increased security since it implements atleast 4 layers of encryption, however the overhead incurred in terms of CPU cycles and turnaround time is significantly higher for the onion framework as compared to the causal chain. The Success Rate (*SR*) is defined as the cognition of malicious activities in the cloud on the basis of the attack model considered here. More specifically, if the amalgamated provenance and cornerstone can be collectively used to detect the malwares in the cloud, then the *SR* of the model increases. In case of missing the detection despite the amalgamation of web cornerstone, the Miss Rate (*MR*) will increase. The Success Rates (*SR*) for structuring and melding the cornerstone data to security keystone provenance of the six specific SaaS applications are found to be 85.0554%, 96.7032%, 98.3871%, 93.9732%, 80.5000% and 84.9257% respectively. The conclusion of this research suggests the effectiveness of the proposed algorithms and scope of improving the scheme to ameliorate provenance analysis of open source cloud computing such as advanced file transfer techniques to be used for *DFI*.

Preface

This thesis is submitted to the Master of Science in Software Engineering (MSSE) Office of the Institute of Information Technology, University of Dhaka, in partial fulfillment to the requirements for the Degree. It contains the work done from January 2013. My supervisor on the research has been Dr. Kazi Muheymin-Us-Sakib, Program Chair, Day Program Office. The thesis has been made solely by the author; some of the viewpoints are based on the research of others, and references to those sources are provided.

This research has been supported and funded by the University Grants Commission, Bangladesh under the University of Dhaka Teachers Research Grant No-Regi/Admn-3/2012-2013/13190 and the Ministry of Science and Technology (MoST) Research Fellowship No-39.012.002.01.03.019.2013-229.

The purpose of engaging in the research was to solve the research issue of ensuring the integrity of cloud provenance and improve Cloud Security Intelligence through the amalgamation of structured provenance and unstructured cornerstone data from the web. The proposition and performance evaluation of the open source keystone provenance securing blueprint and its amalgamation with cornerstone was the decisive factor in this research. The research is aimed to provide tamper-proof provenance to Cloud Digital Forensic Experts (Cloud-DFI) and amalgamate those with latest web data to achieve better security intelligence.

The reason for this research is the experience of the author in the field of cloud computing for over two years. The main inspiration came from the potential of open source cloud computing once it becomes more secured and increase in the trust of the people. The results of the experiments identify the areas where the proposed scheme performs better and also the scope of improvements and future work. The thesis concludes with a discussion of the research and the additional research questions that have come up in the course of this research.

The author conveys gratitude to the supervisor of this research for the continuous support and reviews. At the same time the author is grateful to all the members of the Institute of Information Technology, University of Dhaka and Panacea Systems Ltd, the software industry that provided the hardware for conducting experiments during the research.

Acknowledgments

This thesis is about devising a scheme for provenance security and web cornerstone data amalgamation in open source cloud that highlights exactly my research interest for the last one and half years. It has been mainly possible for the constant effort of my thesis supervisor Dr. Kazi Muheymin-Us-Sakib, whose continuous support has enabled me to attain international publications in this topic. It all began in 2013 when my research supervisor Dr. Sakib advised me to pursue research in cloud computing provenance. It was at Hayestech where I got my first flavors of cloud computing and there was no looking back from then on. I am extremely grateful to Dr. Sakib for lending me this opportunity. From that moment I started to learn about the cloud using Google to good effect. I connected to many people who deal with this technology, electronically and face to face. Since it is impossible for me to thank them all, I will take this opportunity to mention the names of those without whom this study would have never been completed.

Dr. Sakib was my supervisor in this research, and from that moment he has been guiding me in the research field, identifying my mistakes by going through every line of my write ups and my experimental results, investing significant portions of his valuable time to my development. At the same time he was always concerned about keeping my motivation high and maintaining a high spirit within the research team.

I am also very grateful to Dr. Mahbubul Alam Joarder, in his role as the Director of the Institute of Information Technology, University of Dhaka, for allowing me pursue my research in a healthy and competitive atmosphere of IIT. I will also take this opportunity to thank Dr. Mohammad Shoyaib and his panel for the effective feedback during the second presentation of my thesis. This research has been supported and funded by the University Grants Commission, Bangladesh under the University of Dhaka Teachers Research Grant No-Regi/Admn-3/2012-2013/13190 and the Ministry of Science and Technology (MoST) Research Fellowship No-39.012.002.01.03.019.2013-229.

As I said earlier, I first learnt about the cloud at Hayestech. Hence I convey my heartfelt gratitude to Mr. Shaniur TIM Nabi, Chief Technology Officer (CEO) of Hayestech, for keeping faith in me and enabling me to learn the cloud as an intern at his company. Also, my heartiest thanks to Mr. Redwanul Karim Ansari, Chief Executive Officer (CEO) of Panacea Systems Ltd, for appointing me as the Lead Research, Cloud Computing at his company. Hence I was fortunate enough to be provided with the powerful cloud servers of Panacea where I executed the tests and obtained the results of the thesis experiment and also for sponsoring my visit to the United Kingdom in 2013 to attend the INTECH conference.

Finally I would like to thank my family and friends who have directly and indirectly helped me in the research. Special thanks go to my mother and father, who have supported me and provided the logistics that enabled me to visit Madrid, Spain and present our research at the HPCS 2012 conference. My elder brother Nahid was always a support and source of confidence. I would like to thank all the students of MSSE01 batch who are seeking completion of the Master of Sciience in Software Engineering (MSSE) program through their participation in research and projects. Thanks go to the staffs of IIT, who have been a continuous source of help. The members of IIT family have really made this research fun for me and made the moments very enjoyable.

Publications

The motivation to proceed with the research and achieve completion got a tremendous boost when our research papers got accepted in reputed journals and conferences. The publications made during the course of this research have been listed below:-

1. ProvIntSec: A Provenance Cognition Scheme Ensuring the Security and Integrity of Real Life Open Source Cloud. *International Journal of Information Privacy, Security and Integrity (IJIPSI)*. Vol. 1, Issue. 4, 2013, pages: 360–380.

2. TFPaaS: Test-first Performance as a Service to Cloud for Software Tesitng Environment. *International Journal of Web Applications (IJWA)*. Vol. 5, Issue 4, 2014. pages: 153–167.

3. Active-Threaded Algorithms for Provenance Cognition in the Cloud preserving Low Overhead and Fault Tolerance. *8th International Conference on Computer Engineering and Applications 2014 (ICEA)*, pages: 119–126.

4. An Empirical Investigation of Cost-Resource Optimization for running Real-Life Applications in Open Source Cloud. *5th The High Performance Computing Symposium, 2012 (HPCS)*, pages: 718–723.

5. IVRIDIO: Design of a Software Testing Framework to provide Test-first Performance as a Service *3rd International Conference on Innovative Computing Technology, 2013 (INTECH)*, pages: 520–525.

6. A Peer to Peer Resource Provisioning Scheme for Cloud Computing environment using Multi Attribute Utility theory. *3rd International Conference on Innovative Computing Technology, 2013 (INTECH)*, pages: 132–137.

7. Cloud-Niagara: A High Availability and Low Overhead Fault Tolerance Middleware for the Cloud. *16th International Conference on Computer and Information Technology, 2013 (ICCIT)*, pages: 164–170.

8. Watchword-Oriented and Time-Stamped Algorithms for Tamper-Proof Cloud Provenance Cognition. *3rd International Conference on Informatics, Electronics and Vision, 2014 (ICIEV)* (accepted for publication).

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Cloud Computing is the dynamic provisioning of resources from a shared resource pool. Cloud has gained popularity due to its capability to rapidly allocate resources in accordance to the requirements of the users, preserving the desired properties of elasticity and scalability. However, the integrity and security of the cloud-based Software as a Service (SaaS) applications and data need to be ensured for widespread acceptance of cloud services.

1.1 Motivation for the Research

Since cloud customers store their data and use cloud-based applications hosted at the servers of the cloud service provider, they are always concerned about the integrity of those. In this scenario, the trustworthiness of both the cloud service provider and the cloud customer must be ensured [1]. Hence Digital Forensic Investigation (DFI) needs to be conducted regularly for the cloud to achieve trustworthiness [1][4]. Fine grained and coarse grained provenance can be detected and effectively analyzed to be used for DFI [5]. Cloud Provenance are the journal data that provides valuable information about the identities, time and data for various events of cloud applications in addition to the data stored and manipulated on cloud compute servers. However, currently the following issues degrade the effectiveness of the cloud provenance for its use in DFI and motivates the cause for research.

- The detected provenance suffers from tampering since the body storing and maintaining the cloud provenance can manipulate the data. The importance of securing provenance information is manifold taking under consideration the acceptability of those to forensic experts. Literature review shows that research importance has been greatly provided in cloud data security and the issue of securing cloud provenance has been considered to a limited extent [1], [4]. The challenge of ensuring the security of provenance in a dynamic environment such as the cloud through multi-trusted bodies has been unaddressed [1]. Since no significant research has been conducted for securing cloud computing provenance despite the importance of those in *DFI*, the trust of cloud environment cannot be established without ensuring cloud provenance security.

- In addition to provenance security, the auditors and forensic experts require additional information for effective Security Intelligence (*SI*). Taking decisions on the basis of fine grained system provenance alone is not sufficient for cloud forensic investigation [6], [1]. Additional information on malware and nefarious activities in the cloud are required [6]. The additional information can be obtained from web data. However those are in unstructured form and requires validation. Validation and integration of web data with provenance to obtain correct and complete security evaluations during *DFI* has not been addressed.

- Securing cloud provenance through a trusted *Auditor* based framework can be a generalized solution for the entire cloud system since it is capable of securing both system and application level provenance journals. However, the secured provenance must be effectively analyzed during *DFI*. Although the provenance is secured it lacks fresh data that has been validated since the provenance might have been detected and stored months ago. However, for nefarious attacks, the malware are upgraded within days and weeks. In this case, data must be gathered from reliable sources in the web. Validated web sources for fresh data on malware activities will add value of freshness to the existing provenance.

- Freshness of data is highly important for the attack model of nefarious activities considered in this research because it will enable achievement of better DFI using cloud provenance. fresh web data will provide new information and once it is melded with provenance, the information will be enriched and the digital forensic investigator will have latest data at increased integrity. As a result, the investigator will not only have system and application level provenance, but also latest information on nefarious activities will be available when web data are amalgamated with cloud provenance. This will yield better results and more accurate Security Intelligence (*SI*).

Besides being fresh, the web data must not manipulate the integrity of the secured cloud provenance. As a result the validity of the cornerstone data needs to ensured through web user and owner ratings. Melding the data with the existing structured provenance must be done for effective analysis. This can be done using taxonomized web data and amalgamation of the taxonomies with the multidimensional spider schema storing the provenance in the cloud.

Unlike the first contribution, web data amalgamation cannot be a generalized solution. It needs to be specific to the research context. Hence for this research, the context is based on a real life attack model of malwares conducting various nefarious activities in the cloud virtual and physical machines. The attack model involves different malicious entities like Lupper, SSHBruteporce, Ramen, Millen, etc. The goal is to mould web data with secured provenance to obtain additional information on the malicious entities to enable detection of those.

The research challenge is that the cloud malwares are highly unpredictable and volatile in behaviour, thus resulting in fast mutation and incorporation of new attack strategies. The new strategies will be available on the web before those can be found in cloud provenance because web data are updated more frequently and by a large number of web users. Hence updates on the malwares must be made available to the forensic investigators through amalgamation of web cornerstone and cloud provenance. For the mentioned case cloud provenance data are not enough since those do not contain the latest information. Web is a source for obtaining fresh information on malware activities, hence the second research question considered here emphasizes on amalgamation of web data with secured provenance to achieve improved *SI*.

The data obtained from the web are referred to as cornerstone data due to the applicability of those in specific areas of *SI*. The cornerstone data are in unstructured format and must be converted into semi-structured or fully-structured form in order to be used with collected provenance to obtain increased accuracy of security analysis of the cloud system.

1.2 Addressing the Research Questions

Based on the motivation stated in the previous section, the importance for securing provenance in the cloud and the amalgamation of those with cornerstone data for Cloud Security Intelligence (*CSI*) is manifold for preserving the integrity and value of cloud provenance. The criticality of the issue and the lack of research conducted in this field bring up two important research questions as stated below.

1. ***How can detected provenance be secured to feed digital forensic experts with increased volume of tamper proof data?*** Related research have increasingly focused on securing the cloud data at the application layers [7], [6]. However, very minute research efforts have been provided to ensure the security of cloud provenance despite the importance of those in *DFI*. A number of recent research emphasized the need for securing provenance in cloud without providing a feasible solution tested on real life cloud [6]. Although the effectiveness of cloud provenance security is recognized, a generalized model for securing provenance at the application and system layers of the cloud need to be proposed.

2. ***How can the structured provenance data be melded with unstructured cornerstone web data from reliable web sources to yield digital security intelligence with greater accuracy?*** Research has been conducted to a limited scale for capturing, structuring and melding cornerstone data with cloud computing provenance [1] [5]. Although state of the art solutions have been provided to ensure privacy of cloud data as in [5], amalgamation of web cornerstone with provenance keystone has not been conducted to the best of knowledge. The information present in system provenance may not be enough to make *SI* decisions, such as fresh information. Such data are freely available on the web as unstructured, situational cornerstones.

The information present in system provenance may not be enough to make *SI* decisions, such as fresh information. Such data are freely available on the web as unstructured, situational cornerstones. If those are applied in specific situations correctly, the system can be enriched with huge volume of effective data that can provide *SI* to a significant level.

1.3 Research Contributions

Provenance in the cloud can be used as auditing data in DFI. Since a large number of uncertain and low confidence transactions take place in the cloud, provenance can be used to identify the propagation of the transactions and aid forensic investigators to identify the root cause of cloud activities. However, the provenance itself is susceptible to manipulation and tampering if those are not secured. This will result in the lack of confidence to the forensic investigators on cloud provenance. Hence the integrity of the provenance or lineage information itself needs to be ensured, otherwise the forensic investigators will lack reliability on cloud provenance.

Contribution 1: Securing the Provenance Scheme

Securing the provenance is necessary to preserve the integrity of the information for

DFI. The fact cannot be ignored that secured provenance can play an important role in cloud system forensics. The importance of securing provenance in the cloud is manifold since a large number of virtual machine (*vm*) instances and physical machines (*pm*) are involved and the *vm*'s are mapped to specific *pm*'s. Hence a large volume of data are needed to be effectively handled through provenance journals and those must be secured.

The proposed scheme ensures the security of provenance in a series of steps, the first of which involves binding the provenance journal data and user data through which the provenance was derived. Active-threading is a mechanism that checks multiple provenance journals and binds those from the *vm*'s to *pm*'s. Active-threading mechanism is used to treat all provenance journals as individual objects and those are then bounded with multiple user data. Next the bounded provenance are encrypted using the private key of the *ProvenanceOwner* (PO). Next, those are passed to an independent trusted body called the *Auditor*.

The *Auditor* is included to ensure that the *PO* does not have a single point of control. The *Auditor* process is controlled by an organization that is trusted by both the cloud provider and the cloud customer. The cloud system level provenance is under consideration for security. The *Auditor* encrypts the captured provenance using its private key. The public keys of the *PO* and *Auditor* are shared to aid in decryption of the provenance when those are needed to be analyzed during *DFI*. Analysis of results show that 45% of the instances face an average delay of 7-8 seconds due to the application of the proposed mechanism. The delay between 9-9.9 seconds is faced by 40% of the instances for encapsulating provenance meta-file.

Contribution 2: Amalgamation of system level provenance with web cornerstone data

Provenance have been generally referred to as keystone, signifying the criticality of those in *CSI*. On the other hand, web data store are referred to as cornerstone as specified earlier since those only add value in specific situations. The importance of melding cornerstone web data with keystone provenance is important since those provided freshness and large volume of useful information. Hence using provenance for *CSI* can be increasingly effective if the strength of web cornerstone data can be harnessed to increase the information volume.

The proposed methodology provides algorithms that acquire cornerstone data from the web from specific sources using watchword matching scheme [8], [2]. Specific attributes like owner rating, user rating and freshness have been considered as verification criteria for the algorithms to increase the acceptance of data. Next Effective-Descriptive set theory have been proposed for identifying valid web data that treats sources as specific set of elements [2]. Each set of applications consists of a subset of processes from which the cornerstone data are obtained.

Fusion cube based architecture using spider schema has been proposed to meld the cornerstone data with keystone provenance. Multi-dimensional properties of spider schema ensures accurate fusion of data attributes and treats the data objects as dynamic entities. The success of the algorithm is determined on the capability of the proposed model to verify the rating of the web cornerstone data and merging the data with provenance through taxonomy and multi-dimensional spider schema.

The Success Rate (*SR*) is defined as the detection of malware using the provenance and data from the web collectively. Hence, for the defined research threat model, if the provenance information predicate and the web cornerstone data can together identify the malware in the cloud, then the *SR* increases in value. Otherwise it is a case of missing the malware detection and the Miss Rate (*MR*) increases. Throughout the research experiments in real life cloud environment, the *SR* for structuring and melding the cornerstone data to security keystone provenance are found to be 85.0554%, 96.7032%, 98.3871%, 93.9732%, 80.5000% and 84.9257% respectively for the six real life cloud SaaS applications namely

Inventory Management Software (*IM*), Accounting Management Software (*AM*), Human Resource Management Software (*HRM*), Office Document Management Software (*ODM*), Sales Management Software (*SM*), Customer Relationship Management Software (*CRM*).

1.4 Thesis Structure

This section identifies the chapters of the thesis and provides an overview of those. The thesis is divided into 7 chapters, identifying the different contributions made throughout the course of the research.

- **Chapter 2:** This chapter provides an in-depth background study regarding cloud computing provenance. The chapter identifies the mechanisms involved in cloud computing and how system level provenance can be captured from those. At the same time it identifies the real life application areas where cloud provenance can be used.
- **Chapter 3:** An exhaustive literature review based on related research have been conducted and presented in this chapter. It focuses the different areas of provenance research and the current approaches available such as watermarking, time-stamping, etc [8]. Additionally this chapter highlights the information proving that the research questions addressed in this thesis have not been considered in previous work.
- **Chapter 4:** The chapter introduces the novel solution to the first research question. The framework for securing cloud computing provenance through two independent and trusted entities namely *Provenance Owner* (*PO*) and *Auditor* have been identified. Next, algorithms for realizing the proposed solution have been stated. The chapter concludes with an overview of the proposed mechanism's process flow.
- **Chapter 5:** This chapter identifies the solution to merging secured cloud provenance

with fresh and high confidence web data referred to as keystone and cornerstone respectively for improved and accurate cloud security analytics [9]. The proposed architecture is followed by algorithms and mathematical models for crawling, structuring, verifying the confidence levels and melding those with keystone provenance. The chapter ends with the progress flow of the proposed Cloud Security Intelligence (*CSI*) scheme.

- **Chapter 6:** The results obtained for the implementation of two research questions have been highlighted here. The experimental test-bed has been described and the test methods evaluated based on real life production level cloud environment. The obtained results were tabulated and represented for different test scenarios and Software as a Service (SaaS) applications. Graphical analysis of results were conducted and the reasons for the desired performance were identified and stated.

- **Chapter 7:** This chapter concludes the thesis summarizing major research contributions and providing evaluation of the desired results. At the same time the system focuses on the developments of future research on the basis of the current progress.

Chapter 2

Background

In this chapter, the distributed architecture of cloud computing, research assumptions and pre-requisites for the research have been analysed. Cloud has enabled many organizations to carry out complex computation with limited resources which would otherwise be impossible [10]. Cloud is an evolving technology, the basis of which is virtualization of physical servers to form virtual machines (vm) instances [10]. Despite the advancement in cloud, limitations exist in terms of security. However, cloud computing has limitations when it comes to acceptability, accountability, traceability and security. Customers are unwilling to provide their data and computation on the cloud despite the financial benefits, the chief reason for this is the loss of control and the security of tracking the activities which occur with the computation and data. There is a strong requirement to propose effective provenance mechanisms for this distributed and dynamic environment.

In addition to the problems posed by distributed applications, flexibility, representation and management of provenance information present new challenges for the cloud [10]. Log based provenance is required to support those. In order to solve these issues, there is an increasing need for research on file-centric and system-centric provenance detection from simultaneously generated process logs of cloud computing environment. Provenance detection will enable customers to keep track of their data and computation on the cloud.

Existing mechanisms are not suitable for cloud provenance. Ko. et al identified accountability and audibility as prime challenges for cloud computing [10]. Through real life scenarios of malicious cloud attacks, the vulnerability of cloud to attacks was shown [1]. However the research did not focus on how provenance detection can effectively prevent such attacks. Technical and procedural approaches to ensure security of cloud were discussed in [1]. Schemes such as anonymous login and identification of anonymous users through pseudomonas file read were proposed in the research. However, the challenges of the proposed provenance detection approach were not shown in [1].

Specific challenges for provenance detection on the cloud must be addressed. These challenges must highlight log-based provenance tracking for file system logs and identification of platform specific provenance detection. Visual representations of provenance data are an additional challenge. Atomic actions at the system level of cloud and critical cloud process files should be identified and studied to find provenance of those. From analysis of existing research, a rule based provenance scheme must be proposed for provenance capture and representation.

Provenance in cloud computing is of increasing importance, as reflected by numerous research works [11],[12],[13]. Provenance in cloud is different from provenance of other distributed system since the provenance data should find the relationship between the physical machines and the virtual machine instances. This is done to aid cloud administrators identify which physical machine is providing computing and storage resources to which particular virtual machine instance.

2.1 Provenance

Provenance is the critical information to ensure security, reliability and trustworthiness of distributed computer environments [13]. Provenance is metadata that pertains to the derivation history of data artifacts [13]. Artifacts are the data objects, processes, operations, and other activities which are executed by the users and administrators of distributed environments. Information regarding the creation time, the modifier and the process used to modify the data stored or the processes running in distributed environments allows system administrators and data forensic experts to estimate the validity and the reliability of data.

Recognizing the importance of provenance, researchers are concerned with the detection, storage, representation and analysis of the correctness of provenance data from provenance information captured real time or stored in provenance repositories. Many research organizations use provenance information collected from distributed applications to analyze and obtain answers to research questions. Hence many organizations are hosting their provenance data so that research communities can use the data and carry out experiments.

A specialized application within a pipeline of tasks is run in a distributed environment due to resource dependencies, that environment may collect and store provenance for all data products derived by that application. The distributed environment should provide the ability to collect and store provenance for all operations executed at the physical level as well as the virtual level, while provenance collected from other parts of the workflow may go to a different repository. As data sharing across heterogeneous environments becomes more common with multidisciplinary research, the need to query or reassemble provenance from across multiple repositories are becoming increasingly necessary.

2.2 Assumptions

The research is taken forward while keeping in mind certain assumptions and obstacles regarding cloud computing provenance detection. Each problem is paired with an opportunity showing how provenance detection can help solve the issues, ranging from provenance detection to analysis of the captured provenance data.

Security of the cloud is often the most cited objection since customers are severely concerned about the security of the data they store on the cloud. The security issues which are used to protect the cloud from attacks are similar to the ones implemented at large scale data centres, with the exception that both customers and not only cloud administrators are responsible for the security. In addition to the two parties, a third party may be involved for the security of the cloud which includes providers of value added services which are incorporated into the cloud. For example, RightScale provides value added services, a few of which are automatically incorporated with Amazon Elastic Cloud Compute (EC2) and helps dynamic scale up or scale down of EC2.

The cloud users are responsible for the security of the vm instances which are sanctioned. The cloud administrators are responsible for the security of the physical layers, as well as the security of vm layer by monitoring data regarding launching of vm instances and tracking logs of file access and changes made to those. So security of the intermediate layers and vm instances are shared among the cloud administrators and users. If the users are exposed to greater details of vm instances, more responsibilities are handed over from the administrator to the user. Some cloud providers outsource the security maintenance task to third parties who have independent IT management to manage the cloud services. Amazon EC2 users have more technical responsibility than Azure users, who in turn have more technical responsibilities than AppEngine users [11].

The primary security mechanism that can be used in todays cloud infrastructure is log based provenance detection [1]. Log based provenance detection can be used to monitor the vm instances from launching to termination [11]. This is a powerful tool to ensure confidentiality and audibility, since attempts by attackers to deactivate vm instances or stop underlying cloud processes can be identified. Incorrect provenance detection mechanisms can result in attacker having unauthorized access to the customers vm state information. The challenges are similar to large data centres which do not implement cloud computing, where various programs need to be protected from each other physically. Any large data centre, be it cloud or non-cloud, will look to detect provenance of the vm layer to detect malicious activities.

Similarly, audibility must be added as a security mechanism which would include provenance detection of physical data centres, which is beyond the reach of any outsider, be it a user or a malicious attacker. Provenance detection can provide full security to the vm layers and also to the physical layers, and makes the cloud arguably more secured as compared to other vendor-provided security applications which act only at the vm layer. Provenance detection scheme can ensure detection of any unauthorized activity about vm instances as well as the system itself.

2.3 Provenance Overhead

Cloud provenance data is metadata about activities and operations. The amount of computation and storage overhead required for provenance detection is an area of future research interest. How can provenance data be effectively stored is also an active research area. Also an effective storage procedure and storage unit of provenance data should be determined in future research.

A provenance system must be user-friendly from the viewpoint of users and cloud service providers. The research challenge is to device a provenance mechanism that can be easily incorporated and removed from the cloud as wanted by the stakeholders. The goal of this research will be to make the cloud architecture unaffected from the provenance mechanism.

2.4 Limited Availability

Compelling research is required to identify the various cloud computing processes and log the activities of those in order to accurately detect provenance and understand the behaviour of the cloud system. This is necessary to identify any problems that may occur in the cloud administration.

One of the difficult challenges in cloud computing is the limited availability of provenance detection systems for open source cloud. A common occurrence is that these limitations are not present for proprietary cloud services, so most of the research for provenance detection is based on those. One opportunity may be devising a scheme that tracks provenance on the basis of both system and file-centric logs in Cloud Computing. Many open source cloud computing providers developed their infrastructure without using any provenance detection, either because the recent surge in security of cloud through provenance detection was not comprehended or the infrastructure was developed solely for research purpose. Log based provenance detection can capture information of vms which would otherwise been impossible for open source cloud.

Provenance detection can be widely used in data forensics of the cloud to identify the root cause of system failures. However even reputed cloud service providers like Amazon EC2 and Google AppEngine have faced service breaks. With effective provenance detection of logs, data forensic experts have successfully determined the reason of the flaws. Such provenance detection schemes, if developed for open source cloud platforms, can also help detection of reasons for service breaks. Following tables illustrates some of the service disruptions of proprietary cloud services and the reasons identified through provenance information for such failures [12].

Table 2.1: Failures due to lack of provenance data

Service	Reason of Failure	Duration
Amazon S3	Authentication mechanisms overloaded by remote attacks	2 hours
Google AppEngine	Error from the end of the maintenance engineers program	4.6 hours
Gmail	The contact list mechanism crashed	1.4 hours

2.5 Compatibility and Automation

Cloud computing involves a large number of process execution, system calls, file operations within a short time period. It is very important to make the provenance detection mechanism completely automated and stable to handle the large data load.

The provenance detection scheme must be tested to ensure that failure does not occur. The scheme must have the same performance of scalability, reliability and fault tolerance as the cloud itself. In this regard two important testing research that should be carried out are failure testing and failover testing of the cloud.

Open source cloud requires detailed understanding, whereas proprietary clouds have closed source. The aim of this research is to develop a provenance detection model for the cloud that is independent of any platform and fits into any cloud environment.

Since large volume data is captured as provenance in the cloud, research should be carried to present his provenance data in a visual format to the viewers of provenance information. Visual representations of provenance data must be specific for the cloud customers, service providers and regulators since their needs are different. Hence research should be carried out in this regard.

2.6 Provenance in the Cloud

Provenance is information which outlines the history of an artefact or object. It must be mentioned that provenance is an essential component for data stored on the cloud and properties of provenance that enable its utility should be identify. Current cloud offerings and design to implement protocols for maintaining data/provenance in cloud stores has to be identified. The protocols should represent different points in the design space and satisfy different subsets of the provenance properties. One can select a protocol based upon the provided properties without sacrificing performance.

Despite the feasibility to provide provenance as an additional layer on top of the existing cloud, provenance should be incorporated in the cloud as a core cloud feature. Traditional work in storage and file systems addresses the storing of information and making it available to users. Provenance addresses that correct information are made available to the customers at the correct time. Provenance, sometimes called lineage, is metadata detailing the derivation of an object. If it were possible to fully capture provenance for digital documents and transactions, detecting insider trading, reproducing research results, and identifying the source of system break-ins would be easy [12].

Provenance is particularly crucial in the cloud, because data in the cloud can be shared widely and anonymously. Without provenance, data consumers have no means to verify its authenticity or identity. The web has taught us that widely shared, easy-to-publish data are useful, but it has also taught us to be sceptical consumers. For shared data on the cloud, it is impossible to know exactly how updated or trustworthy are the data on the web.

2.7 Attributes of Provenance Systems for the Cloud

Provenance for the cloud must ensure that the access time, manipulation type and accessory are correctly identified. Following are some essential attributes which satisfy the stated capabilities.

1. ***Provenance Data Coupling:*** : The data-coupling property states that an object and its provenance must match that is, the provenance must accurately and completely describe the data. This property allows users to make accurate decisions using provenance. Without data-coupling, a client might use old data based on new provenance or might use new data based on old provenance. In both of these cases, the user relying on the provenance is misled into using invalid data. Systems that do not provide data-coupling during writes can detect data-coupling violations on access and with-

hold or explicitly identify objects without accurate provenance. For example, if the provenance includes a hash of the data, we can compute the hash of a data item to determine if its provenance refers to this version of that data.

Detection is, at best, a mediocre replacement for data-coupling, because although users will not be misled, they cannot safely use available data when its provenance is wrong.

2. ***Eventual data coupling:*** In eventual data coupling, the data and its provenance might not be consistent at a particular instant, but are guaranteed to be eventually match. With eventual data-coupling, a system requires detection, since there may exist intervals during which an object and its provenance do not match.

3. ***Causal Ordering:*** This property acknowledges the causal relationship among objects. If an object O, is the result of transforming input data P, then the provenance of O is the super-set of the provenance of P. Thus, a system must ensure that an objects ancestors (and their provenance) are persistent before making the object itself persistent. Multi-object Causal Ordering violations occur when the system writes an object to persistent store before writing all its ancestors, and the system crashes before recording those ancestors and their provenance. Similar to eventual data coupling, a weaker form of the property Eventual Causal Ordering is realizable. A system still requires detection to account for the intervals during which an objects provenance may be incomplete, because its ancestors and their provenance are not yet persistent or not available due to eventual consistency.

4. ***Data-Independent Persistence:*** This property ensures that a system retains an objects provenance, even if the object is removed. As in the last section, assume that P is an ancestor of O. If P were removed, Os provenance still includes the provenance of P, so a system must make sure to retain Ps provenance, even if P no longer ex-

ists. If *P*s provenance is deleted when *P* is deleted, parts of the provenance Directed Acyclic Graph (*DAG*) will become disconnected. If *P* had no descendants, then a system might choose to remove its provenance, since it would no longer be accessible via any provenance chain. Another approach to solving this problem is to copy and propagate an ancestors provenance to its descendants. This is inefficient in terms of space and can quickly become unwieldy.

Since provenance is created more frequently than it is queried, efficient provenance recording is essential. However, efficient query is also important as provenance must be accessible to users who want to access or verify provenance properties of their data.

In scenarios where the number of objects is few or users already know the objects whose provenance they want to access, efficiency is not an issue. Efficiency matters, however, when the number of objects is sizeable and users are unsure of the objects they want to access. For example, users might want to retrieve objects whose provenance matches certain criteria. In cases such as this, if a system stores provenance, but that provenance is not easily queried, the provenance is of reduced value.

High rates of FP will result in increased workload in analyzing and responding to events. They may also result in reduced productivity due to the prevention of legitimate documents and messages from reaching employees.

As with other security measures, a high rate of false negatives will lead to a false sense of security, plus potentially placing the organization in jeopardy from confidential data which is leaked without being identified. At the same time, provenance data should have the ability to detect data flooding, file type/format manipulation.

2.8 Critical Scheduling of Cloud Systems

Scheduling in cloud is a critical process since it defines the specific resource allocation and operation [14]. It runs as a daemon named nova-schedule for OpenStack Essex and picks up a compute server from a pool of available resources depending on the scheduling algorithm in place.

1. ***No prior knowledge about incoming processes:*** A good process scheduling algorithm of distributed systems should work with no prior knowledge of the processes or operations which are to be executed. Algorithms which function based on prior information pose an additional burden on the users since the users then must explicitly specify the characteristics of the operation they submit.

2. ***Dynamic load balancing capability:*** The algorithm should have the capability of managing the dynamically changing load of the nodes. Process assignment on different nodes should be based on the current load status of the nodes and not on some pre specified policy.

3. ***Thrashing Resistant:*** Processor thrashing occurs when the algorithm causes all the nodes of the system to spend all their time migrating processes without accomplishing any useful work.

4. ***Scalable:*** The algorithm should be capable of supporting a large number of nodes from a small number without computational complexity. The attribute must sustain even when the number of nodes is suddenly reduced.

5. ***Fault Tolerant:*** The algorithm should not suffer from crash of a single node of the system. The system should continue functioning with the available nodes which are up at that point of time. The cloud controller should be capable of provisioning

resources from its own physical machines which would increase the fault tolerance capability of the system.

6. ***Quick decision making:*** The scheduling algorithm must exhibit good decision making capability. This is very important at times when the computational requirements of the processes are very high. For example, a customer executing scientific calculations on the instances may require sudden requirement of storage. The algorithm must be able to provide the required storage without termination of the instances.

A scheduler can base its decisions on various factors such as load, memory, physical distance of the availability zone, CPU architecture, etc. The nova scheduler implements a pluggable architecture.

2.9 Critical Security Process Identified for Distributed Applications

Provenance information should be able to thwart a number of security breaches in the distributed environment. The cloud computing major security concern need to be listed and the attacks which can be detected through the analysis of provenance information must be identified at an early stage of the research. There are two approaches to management of malicious cloud activities which are discussed below.

1. ***Preventive approach:*** Preventive approach includes prevention of attacks through implementation of security features at the physical levels. These include installing firewalls and network filters to prevent attacks from being made [15].

2. ***Detective approach:*** The detective approach involves implementation of methods to detect and monitor potential malicious activities. The activities include detection of

data leakage, any misbehaviour in the administration of the cloud and detection of potential attacks on the basis of provenance data collected from log files [15].

3. ***Estimation of risk levels:*** The level of risk is estimated on the basis of the likelihood of an incident scenario, mapped against the estimated negative impact. The likelihood of an incident scenario is given by a threat exploiting vulnerability with a given likelihood. The likelihood of each incident scenario and the business impact was determined in consultation with the expert group contributing to this report, drawing on their collective experience. In cases where it was judged not possible to provide a well-funded estimation of the likelihood of an occurrence, the value is non-applicable. In many cases the estimate of likelihood depends heavily on the cloud model or architecture under consideration. The following shows the risk level as a function of the business impact and likelihood of the incident scenario. The resulting risk is measured on a scale of 0 to 8 that can be evaluated against risk acceptance criteria. This risk scale could also be mapped to a simple overall risk rating.

   (a) ***Low risk:*** 0-2

   (b) ***Medium risk:*** 3-5

   (c) ***High Risk:*** 6-8

   Cloud services are not only about convenient storage, accessible by multiple devices, but include important benefits such as more convenient communication and instant multi-point collaboration. Therefore, a comparative analysis needs to compare not only the risks of storing data in different places (on premises versus the cloud) but also the risks when on premises-data stored on premises, for example, a spreadsheet - is emailed to other persons for their contributions, against the security issues of a spreadsheet stored in the cloud and open to collaboration between those persons.

Therefore, the risks of using cloud computing should be compared to the risks of staying with traditional solutions, such as desktop-based models.

Chapter 3

Literature Review

In this chapter, the existing research issues in secured provenance, secure access and graphical representation of provenance information across multiple processes have been highlighted. At the same time the scope of research in this growing field are highlighted. With regard to the rapid growth of cloud, obtaining and maintaining secured provenance information will ensure that the data stored on the cloud is reliable [4], [5].

Existing research highlight that provenance information based on system and operation logs will ensure the integrity of cloud data [4]. Preliminary studies of the computer forensic community in the field of digital forensics have to be revised and adapted to the environment for effective analysis of provenance information. Investigators need the possibility of reconstructing the corresponding environment for recreating scenarios and test hypothesizes through representation of provenance information. Hence the scope of work in this area is manifold. Following sections identifies how security of provenance information is ensured in current research.

3.1 Security of Provenance Information

Provenance is metadata that is used to identify historical information about an activity. System provenance can be used to identify the date, time and method of activities executed on different processes. Hence, provenance information can be used for digital investigation to detect malicious actions.

Provenance information from individual machines is useless if the interrelationships are not established, since the interconnection between the virtual and physical machines cannot be achieved. Hence the source of the malicious act on the cloud cannot be identified.

Following research contributions have emphasized the need of chaining system provenance from distributed system and addressed the need to secure the provenance itself.

3.1.1 Chaining for Securing Provenance

The authors in [4] and [6] addressed the fact that significant research have been conducted for provenance usage information to reach decisions. A public key framework to ensure the integrity, confidentiality and availability of provenance information via encryption of provenance data before passing those over the network is discussed in [4]. Every user was assumed to have a public/private key pair and the public key of each receiver was known to their respective senders [4]. Auditors were used to verify the trusted connection among different nodes to ensure integrity.

The major contributions by the authors included comparison of the performance of the proposed framework with other provenance detection schemes [4]. At the same time provenance overhead was calculated and compared with existing solutions for performance analysis. Analysis of provenance with security schemes detecting malicious activities from those were considered to a limited extent.

3.1.2 Autonomous Reliability Aware Negotiation (ARAN)

Automating the cloud service offering using provenance on the cloud is an important security contribution since it reduces human intervention [16]. The authors proposed a state of the art solution to automate the negotiation process of cloud service offering in [16]. They stated that in their previous work the solution was based on the trust of the Quality of Service by the providers [16]. They argued that clients should be able to process the reliability of the offers since they are paying for the services. Hence the authors felt it is important to provide an automated negotiation model that takes the opinion of the users into account.

The model included negotiation services providing feedback to the data center monitoring for automating virtual machine (vm) instance allocation. The client end of the negotiation service involved taking requests and opinions from the users. Ontology based discovery using Web Service Modelling Language (WSML) is used to take in user opinions. The Cloud service repository feeds in information about the offerings to the users. The semantic Service Level Agreement (SLA) repository is synced with the third party cloud monitoring service to ensure proper audability and trust of the users [16]. The contributions of the authors is that they aimed to monitor to maximize the availability of the cloud service and reduce cost. In addition, the system manages the ensure trust assurance of the negotiation of service by incorporating both the customers and users in the process [16]. Adding provenance to the existing automated system to identify flaws and security threats in the SLA through meta-data will increase the strength of the framework.

3.1.3 The Probabilistic Provenance Graph

Real life motivating examples were provided to show the uncertainty in provenance information in [17]. The authors dealt with provenance uncertainty using Probabilistic Provenance Graphs (PPG) to represent the uncertain situations [17]. Provenance information is generally considered certain since it is generated by system itself with minimal human intervention. Two examples involved determination of spam and authentic email messages. PPG was used to represent uncertain provenance scenarios with different core provenance relationships. PPG is a directed acyclic graph where the vertices represent the artifact, event or activity [17]. The edges correspond to relationship functions and events. The relationship function represent the type of events that are associated with each edge.

It was assumed that only edges from the same parent and same type may belong to the same sample space. Partial functions are associated to model the relationship between the model and beliefs in the assignment of the relationship functions. The contribution involves probabilistic extensions of the Open Provenance Model through integration of PPG in provenance modelling. PPG is a novel approach of representing provenance that was proposed here. Finally a meta-algorithm to find the linguistic relationships among provenance information was proposed. Modelling PPG to identify and color provenance data about malware was not considered.

3.1.4 Formal Framework for Provenance Security

The authors outlined a provenance framework that includes a model ensuring the security of provenance data through topological analysis [18], [19]. Disclosure and Obfuscation were identified as security policies for provenance. Implications of provenance in the field of automata, workflow provenance graphs and database queries are explored here [18]. Formal definitions and implications of provenance are important since the existing researches focused mainly on decision properties that arise from specific form of data as stated by the authors in [18]. General form of provenance are yet to be defined for the cloud since a large number of variety of systems are involved here [18]. Specific forms of provenance such as experimental workflow provenance are represented using Directed Acyclic Graphs (DAG). However, research needs to be done to make the information motivations formal.

The assumption was based on a number of principals that are equipped with a set of possible behaviors called traces. Traces are considered to contain all the relevant information about a system that may be possibly required for scientific workflows or computer forensic [20]. Since this is a subjective judgment, therefore design of traces and storage of those need to be verified by system analysts as highlighted in [20]. Traces are identified as the general form of provenance as far as the principles interacting with it are concerned.

The authors stated that provenance will be loosely specified if conventional techniques are used to securing provenance. Disclosure is defined as ensuring that a given principal will always satisfy a given provenance retrieval [20]. Obfuscation is a principal that can never certainly deduce information from provenance query. The experiment tested the advertised ability of popular tools to collect forensic data remotely in the cloud at the guest Operating System.

Success or failure would be measured if the tool was able to collect evidence remotely, and how accurately the data compared to those from a standalone control machine. The main contributions involved framework for provenance and the definitions [20]. The generality of the definitions are proven using finite state automata and examples of transducers. However, examples of graphs to chain provenance information from activity, operating systems and kernel layers have not been provided. Also making the provenance non-modifiable have been considered to a limited extent.

Identification of provenance attributes and obstacles to securing provenance information in the cloud is necessary for user-friendliness and effectiveness of provenance detection. Also research in the field of provenance will ease cloud maintainability and at the same time develop user confidence. Regarding digital forensics, the loss of control caused by Cloud environments and vendors presents a huge challenge for investigators.

3.2 Provenance Tracing: Log Analysis

System and data logs have been analyzed for a number of purposes, starting from error identification to data forensic. Provenance information can be traced using log files regarding system access, time of access, processed executed and data manipulated. The role of log analysis for provenance detection in a dynamically scaling system such as the cloud is discussed in this section.

3.2.1 Analysis of Tracer Data

In [21] the authors tried to understand the characteristics of workloads running in MapReduce environments through provenance analysis. It was addressed that understanding the characteristics of workloads for different scientific applications running on the cloud will be beneficial for both the cloud service provider and cloud customer. The cloud service providers can use this information to better analyze specific workloads and make effective resource provisions for those. The customers can comprehend the factors that affect the performance. Also users can analyze the cloud resources that are important for those. For experimental purpose, ten months of provenance trace data from M45 server in Yahoo was analyzed in [21]. The system runs Hadoop and provides free cloud resource for system research at selected universities.

The M45 cluster consists of approximately 400 nodes, 4000 processors, 3 terabytes of memory, and 1.5 petabytes of disk space. The clusters runs Hadoop and provisions small virtual clusters over a large physical cluster [21]. Statistical analysis of the server performance based on provenance data was provided. In addition, analysis of provenance logs identified the performance and failure characteristics of Hadoop jobs running in large scale real life clusters. The identification of security vulnerabilities and malicious activities was not done by the provenance detection scheme. At the same time the provenance results did not yield any performance degradation caused by malware actions.

3.2.2 Visual Log-Analysis

Analyze logs of electronic mail to identify information about Map and Reduce (MR) programs at the MR level abstraction of Hadoop was proposed in [22]. The proposed algorithm helps to understand and analyze Hadoop jobs in real life clusters. In addition performance benchmarks and overheads for different tasks in the cluster have been evaluated. The proposed algorithm extracts local node centric Hadoop provenance data. Next the obtained data are correlated among different nodes and processed to obtain performance information of job structures in Hadoop [22].

The analysis result yields a unique end to end representation of causal activities in the cluster also known as Job-Centric Data-Flow (JCDF). State machine abstractions for Hadoop execution was used in the methodology. However, such mechanism cannot be used to detect vulnerabilities like SlowLoris and Code Injection Attacks in the cloud.

3.2.3 Automatic Capture and Reconstruction of Computational Provenance

The authors aimed to extract provenance information from arbitrary applications through monitoring their transactions with the environment in [23] since browsers can effectively accessed from mobile and desktop environments. It is important to address the fact that scientific research are not found to maintain tracer logs of the communication of the software with a large number of environmental factors. The authors used passive monitoring, overriding and instrumentation techniques for identifying provenance from applications through their socket level communication with different environmental aspects [23]. The real life system called Earth System Science Server (ESSE) software components monitored the activities and provenance were collected from those. Provenance graphs are generated from the obtained information.

The chief contribution addressed here involved the advantage that researchers gain when they see what external environmental factors have impact on their software. This enables researchers to take precise controls to limit the effect of external environment on the scientific research [23]. Real life proprietary software suffer from large number of malicious attacks. Analysis of the provenance graphs to identify external threats to the software applications have not been considered.

3.2.4 Displaying and Querying provenance through Provenance Browsers

The authors proposed a provenance browser framework for querying and visualizing provenance information for scientific workflows [24]. The proposed browser enables users to view complex workflows of experiments through provenance graphs. Hence process monitoring is achieved in a user friendly manner. The authors designed provenance browser architecture with high level queries to retrieve provenance information from PostgreSQL database [24].

Through effective query optimization techniques of the proposed methodology, complex provenance queries can be displayed at a shorter time compared to traditional provenance visualization schemes. However the determination of malicious entities in the cloud through provenance schemes have not been considered.

3.2.5 Limiting Privacy Breaches using Data Provenance

Devising a mechanism to illicit provenance in a tamper proof method is proposed in [22]. The authors stated that recent efforts in data forensic is based on kernel resident codes that track changes to file systems [22]. However most of the approaches are not properly isolated due to additional cost of maintenance and isolation, hence those result in tampering. The integrity of the data from such approaches cannot be guaranteed. The methodology provided here investigates accesses of object to disks and follows the causal chain of accesses across processes, even when the objects are placed in memory [22]. This layer records transaction in a tamper proof audit log.

The log is protected from tampering through controlled access to the files. The primary contribution of the authors is to provide a scheme that captures data accesses to disk and subsequent accesses of that data in memory [22]. The system achieves the above without any monitoring code in the virtual machine and embraces the abstraction of the hypervisor. However, despite being tamper proof how to control the security of the captured provenance in a decentralized environment have been conducted to a limited extent.

3.2.6 Provenance-Aware Tracing of Worm Break-in and Contamination

Process aware approach to worm attacks and contamination have been designed, implemented and evaluated in [25]. The authors identified the important issue that provenance un-awareness leads to problems in quick and accurate identification of worm attack points [25].

Process coloring can be assigned to uniquely identify processes, it is inherited by the child processes and diffused through process actions. Process coloring enables fast identification of worm break in point and naturally partitions log data based on colors, thereby ensuring rapid identification [25]. The main contribution of the authors is to effectively reduce the volume of log data that need to be analyzed for worm identification. How to securely transmit the information in a tamper proof method has not been addressed.

3.2.7 Provenance-based Indexing

Connection between microblog messages have been captured and represented graphically in [26]. The authors stated that it is a significant challenge to understand contents for micro blogs due to their short length and dynamic nature. The authors identified the problems for micro blogs regarding traditional keyword search such as shorter length and high noise level. Next, they proposed a provenance based linking of micro-blogs through indexing means called temporal grouping [26].

Temporal grouping is a connection discovery technique that uses temporal relationships of messages. Messages that are related to each other can be extracted on the basis of their temporal relevance and commonality. The chief contributions included using provenance information for extracting contents from previous messages, development and feedback exploration and judging credibility of information [26]. However, the usefulness and challenges of developing a provenance based framework for malware detection in the cloud with trace back was not addressed.

3.2.8 *Provenance Supporting Reproducibility and Comparison*

Open Source Library (OSL) has been proposed to detect different investigation on same raw digital data using a combinatorial number of tool chains. The goal of this approach is to improve the reproducibility and comparison of digital forensic evidence [17]. Simple canonical description of digital evidence provenance that explicitly states the set of tools and transformations that led from acquired raw data to the resulting product have been discussed. The authors argued that existing research on opensource forensic investigation applications are concerned mainly at the network level data access. They believed that an increased concentration on data provenance will make digital forensic investigation more reliable since the changes, date of access, the owner of the data file and the user that accessed the data can be identified [17].

Cryptographic hash of entire disk image was used to ensure no changes to data have occurred from the time of attack until acquisition of evidence. It was assume that images have been acquired properly, or at least that those were not corrupted. It was assumed that tools are combined honestly and in a best effort to meet known methods of analysis, but it was do not assume they are reliable [17]. That is, if a tool gives a record as output, it does not imply it has so correctly. However, the toolswere run without intentional faults. Finally, it was assumed that the specifications rules for parsing data are deterministic and not open for interpretation.

3.3 Securing critical cloud data

Ensuring security of cloud data is a key issue towards gaining widespread acceptance the cloud. Despite repeated efforts in ensuring the primary user data stored on the cloud, the importance of securing meta-data has been considered to a limited extent. This section illustrates the traits in cloud data security and identifies the lack of study conducted in securing provenance data.

3.3.1 Security Issues in Service Delivery Models of Cloud Computing

Identifying security is an important issue that decelerates the widespread growth of cloud computing [14]. Complications for data privacy and data protection have plagued the market of the cloud [14]. The assurance of data integrity is a necessity to ensure acceptance of cloud services in all sectors [27].

The authors suggested a new model that extends the existing model. However, the new model should not threaten or risk the important features of the existing models. Also, it was stated that in cloud services users need to be vigilant about the security breaches that can occur [14]. The authors have identified critical security issues due to the nature of service delivery model in the cloud. They have contributed to this field by identifying critical research questions regarding cloud security. The authors did not identify any malware related vulnerability, neither their survey brought about any real life scenarios.

3.3.2 Cloud-Based Security Vulnerability Assessment

The different security and vulnerability assessment in cloud computing environment have been assessed in [27]. Experiments were carried out in three environments in which the tools reside on the same lab where the targets are, resided not the same lab but on the same campus, and beside off-campus as stated in [27]. Suitable scan tools are applied to the servers to assess what tests were applicable. Next the captured vulnerabilities were collected and assessed for risk.

Risk was determined through the relative assessment to the facility in terms of the expected effect on each critical asset [27]. The authors provided experimental evidence on cloud computing on the basis of centralized and de-centralized architectures. The scope of identifying threats through analysis of provenance data have been covered to a limited extent.

3.3.3 Challenges for Provenance in Cloud Computing

Provenance capturing in the cloud presents significant challenges du to the distributed nature of the cloud and the integration of cloud services with non-cloud ones [28].Through the mitigation of the research issue, the authors stated that it is possible to provide a new provenance detection model for the cloud for better auditing, debugging, billing and forensics. Existing provenance models were studied and means of collecting tracer data from physical and logical levels of the cloud were identified. Next the scope of improving the existing schemes from technical point of view were highlighted in the research challenges. Critical research issues in cloud computing provenance detection for forensics and auditing were identified. Hence the paper looks to harden the security and integrity of cloud computing.

The authors identified that provenance detection for the cloud have been conducted to a limited extent and stated the important need for protocol design that enable provenance detection [28]. The authors stated that provenance detection for the cloud is vital for identifying the access to the cloud data. The authors proposed three protocols that enable provenance detection in both virtual and physical machines in the cloud. In addition the performance of the three protocols were found to be comparable with each other on reasonable terms [28]. Benchmarks of EC2 performance and UML machines running on EC2 were used. The authors provided protocols to incorporate provenance as a core object of cloud. However the process of identifying threats and security vulnerabilities through analysis of captured provenance have not been taken into account.

3.4 Provenance Tracking

Beside detection of provenance, the tracking of meta-data is an important issue to achieve security. Most of the method for tracking provenance for both centralized and distributed systems are rule based. This section highlights the current rule-based approaches towards keeping track of provenance for cloud.

3.4.1 Rule-Based Data Provenance Tracing Algorithms

Tracking of cloud data provenance from initiation to completion of a process is discussed in [11]. Tracking of end to end cloud data computing provenance will enable full accountability and transparency. Rule based provenance detection algorithms were proposed that analyzed various leakages of data communications in the cloud. A novel breakthrough that is crucial for the trust and security of complex computer systems and communication networks is identified in [11].

The forensic algorithms included file copying, file movement detection, file renaming capture through monitoring of process provenance. Keywords such as move(**mv**), copy(**cp**), remove(**rm**) in Linux acted as trigger to activate the model. Cross machine data leakage problems were identified through socket based provenance. However the algorithms are incapable of detecting system errors and also certain atomic actions such as file overriding [11].

3.4.2 Linked Provenance Data

The authors in [29] provided a solution for evaluating inter-operability issues for re-using Open Provenance Model (OPM) based workflow traces. The authors emphasized that increasing inter-operability needs of provenance data that is highly demanded by database systems, World Wide Web (WWW), system monitoring and workflow systems.

Inter-operability issues that arose in Third Provenance Challenge (PC3) were analyzed and based on that the authors identified provenance information increasingly requires generic provenance and domain specific data for future reuse [29]. Emphasis was placed on data linkage to enable open and transparent infrastructure for provenance data reuse. The importance of provenance data reuse for detection of malicious entities in the cloud has been considered to a limited extent.

3.4.3 Managing Authorization Provenance

The authors extending the work on Belief and Trust (BT) to incorporate provenance authorization denoted by D [30]. Hence the resulting methodology is called DBT. The authors stated that existing access control techniques provided few emphasis on provenance based authorization.

The authors devised a set of well-formed formulae based on primitive propositions. Modal operators for specific agents were used to establish trust for identification and access verification. A real life case study was provided to ensure the proposed policy base enforces access control mechanisms working as the root of trust [30]. The authors showed that the capability to manage the provenance is necessary for security purposes such as data breaches, unauthorized access and executing malicious codes. How to securely maintain the provenance without any tampering is a field of research not addressed.

3.4.4 Supporting Information Assurance in Multi-level Secure Environment

Provenance can be captured from invariant messages in an XML based Service Oriented Architecture (SOA) [31]. The authors re-instated the fact that provenance capturing is required for information assurance, availability, integrity and confidentiality of data. The authors believed that a trust system can be built up in an environment where experiments are repeatable. The authors have chosen a multi-level secure environment where it is not always possible to obtain details of all information across security boundaries. In this regard capturing provenance from invariant messages can ensure integrity and safety of information and workflows [31].

The system is designed to work with both peer to peer and message workflow services since in SOA clients directly talk to the SOA servers to obtain the services as SOAP and REST protocols are used for communication [31]. The dynamic routing service used in the framework supports both explicit and role-based destinations. The importance of chaining provenance information for largely distributed systems and ensuring tamper proof provenance have been considered to a limited extent.

3.4.5 Accountability as a Service for the Cloud

Trustworthy Service oriented Architecture (TSOA) was proposed to ensure accountability of the cloud service providers [32]. Provenance was a part of TSOA that ensured the integrity of the customer data stored on the cloud. The authors stated that little effort is provided for detective measures of unwanted activities in the cloud since composed services span across several administration domains, different operating systems that have own interests, functionalities and priorities. Detective mechanism that ensured accountability through provenance detections proposed in [32]. The request, response, state snapshots were detected as part of provenance. Token receipt and challenge response protocols between the cloud servers and users were monitored to detect the requests and responses.

The authors used Amazon EC2 to simulate a real life production event and log the provenance through monitoring the above processes. The chief contribution is the provenance detection scheme. Cost analysis of the proposed TSOA showed that the main cost came from the accountability system and operating system communication. Negligible delay was observed in the case of validating the value of a token against the accountability system in the challenge response protocol [32]. Chaining and watermarking of the captured provenance was not considered here. Also making the provenance tamper proof through cryptography was considered to a limited extent.

3.5 Ensuring Security in Interprocess, Interinstance or Intersystem Communication

Communication between peer-to-peer processes often needs to be traced for data forensic purposes. Ensuring proper monitoring and capturing of system and provenance requires the captured data to be secured as well to prevent tampering and forging. This section identifies the current methodologies for ensuring security of inter process communication.

3.5.1 File-centric Logger for Monitoring File Access and Transfers within Cloud Computing Environments

The authors proposed Flogger that is a novel file-centric logger suitable for accessing logs in both private and public cloud environments [8]. The authors stated that through logging of file accesses and transfers, trust in the cloud can be ensured. The authors proposed a model that records file centric accesses and transfer information from the kernel levels of both private and public clouds.

Through ensuring kernel level recording of file access logs, total transparency and accountability can be ensured in the cloud [8]. Hence Flogger would work to increase the trust of the people on the cloud. The process of obtaining critical information from the kernel level logs to make inference on malicious activities have not been stated.

3.5.2 Recording Process Documentation

The paper proposes protocols for recording provenance for distributed process execution [7]. Recording and documentation of provenance for distributed applications is required to ensure repeatability of the processes. The authors identified critical requirements for provenance data such as provenance are required to be immutable, autonomously creatable and finalizable.

After definition of provenance characteristics, a distributed protocol for provenance documentation was described in [7]. The experiments reveal the desirable performance of the proposed protocols, hence ensuring that record of distributed system provenance can be achieved. However, the effectiveness of the protocol in terms of detecting malicious processes in distributed systems through provenance have been considered to a limited extent.

3.5.3 Data-Provenance Verification for Secure Hosts

Define security property for provenance that ensures that the obtained provenance cannot be tampered [33]. A cryptographic provenance verification approach is provided that ensures data integrity at the kernel level. In addition Trusted Computing Platform protocol is proposed that prevented fake key events from malware through reasonable assumptions.

The authors believed that the stated goal can be achieved through cryptographic management of provenance. The authors proposed a data provenance integrity technique where the source of provenance can be verified [33]. A trust agent in the kernel module marks each keystroke event. The keystoke is shared between the remote server and trust agent. During the relay of the keystoke, the proposed framework monitors the events through provenance detection. Public and Private keys are created that are obtained from the provenance information and matched for integrity using a keyed hash function [33]. If the signatures do not match, the key is not accepted as a valid input. Robust host based traffic monitoring and key integration services were considered as major contributions. Although the system ensures tamper proof provenance transmission, it does not provide a means of documenting provenance for malware detection.

3.5.4 *Cloud Security with Virtualized Defense*

Reputation systems are proposed in [34] to integrate virtual clusters and distributed data centers. A hierarchy of Peer to Peer (P2P) reputation architecture is proposed to protect datacenters at the site level and safeguard stored information at the file access level. The authors believed that trusted P2P data protection will enable wider acceptance of cloud applications like web mail services, online storage of personal information, etc.

The authors advocated for trust management protocols to secure data centers at the network layer to ensure the integrity of data stored in those. At the top of the three level overlay layer consisted of security precautions for worm containment, the middle layer consisted of intrusion prevention detection at the communication level and the bottom layer ensured access to control of data to only authorized users. In all three layers, provenance was used detect unwanted activities and associate specific reputations. The authors extended the reputation approached of Li, et al and Reddy. Et al into the cloud [34], [35]. Hence reputation based provenance information was introduced here. However, the mathematical evaluation of the proposed framework was not provided. In addition, associating cloud provenance to the reputation architecture will contribute to the strength of the framework.

3.5.5 *Provenance Management System for Scientific Workflows*

A provenance collector from the activity layer was proposed in [35]. The authors collected provenance from distributed and heterogeneous environments without affecting the adaptations of the test cases. The authors stated that collecting provenance from a distributed, peer to peer, heterogeneous environment is a significant challenge mainly due to the differing environments that exist in distributed systems. The authors identified three stages of provenance management that includes provenance mechanisms configuration, provenance gathering and provenance analysis [35].

The mentioned kernel level, operating system level and activity level to collect provenance information. For provenance analysis, the core of the model contained intelligent agents and knowledge base to translate and deduce information from provenance. The authors provided an API servicing layer to capture prospective and retrospective provenance. In accordance to the workflow activity distribution, their API is capable of publishing provenance information using web Services. However capturing provenance for security purposes like malware detection have not been taken into account [35]. In addition, no effort was provided to represent the captured provenance in graphical formal. Watermarking the captured provenance will contribute towards achieving the broader goal of the authors.

3.5.6 Accountability and Trust in Cloud Computing

A framework highlighting the requirements of ensuring trust in the cloud was provided in [36]. The provenance requirements for ensuring privacy was highlighted in the proposed model. The authors identified seven security issues in the cloud and stated the importance of ensuring trust in the cloud for widespread acceptance of the technology. They felt it was important to propose a framework that can solve problems of abuse and nefarious activities, insecure application programming and malicious insiders in the cloud. On the basis of the cloud accountability lifecycle, the proposed framework constitute three layers that are system, data and workflow [36].

File centric and operating system level provenance should be collected from the system layers. The aim of collecting data from the system layer is to achieve replay of a critical snapshot. Provenance logger collects tracer logs from data layer since the authors addressed the need to collect history of data. Effectively managing the sheer amount of provenance data was addressed as a challenge. A consistency logger was also proposed to permit replay, backup and restoring of data [36].

The authors contributed to the research of provenance detection by identifying key provenance qualities such as data coupling, multi-object causal ordering, data independent persistence and effective querying of provenance data [36]. The authors did not provide any implementation or mathematical evaluation of their proposed model. At the same time the importance of chaining and watermarking provenance information was not considered. Moreover, ensuring tamper proofing of provenance information was not focused.

3.5.7 Acquiring Forensic Evidence from Cloud Servers

Technical and trust issues that arise during forensic investigation in acquiring evidence for IaaS services have been exposed and explored [37]. The inaccuracy and inadequacy of the current tools to be used for forensic investigation in the cloud have been identified. Keeping data acquisition as the main focus, crimes that target or use IaaS of cloud will emerge in the landscape and investigators will rely on existing tools like EnCase for acquiring information [37]. However, the dynamic nature of data storage in the cloud makes the tools inadequate, catering the need for new and improved investigation applications. The authors exercised three experiments that collected persistent and non-persistent data from Amazon EC2 public cloud [37].

The highlight of the finding was that the tools relied heavily on the trust of the operating systems in the physical machines and virtual machines. The authors provided six layers of data acquisition from IaaS namely Guest application, Guest OS, Virtualization, Host OS, Physical Hardware, Network. For each experiment, a non-cloud based standalone control machine was used to evaluate the success of the test. The control was a Dell workstation with 32-bit Windows 2008 R2, a single 30 GB disk drive and 2 GB RAM. The machine was connected to the Internet and installed the Apache web server. Several web pages with identifying names and content were created [37]. The machines were artificially compromised using a web-based vulnerability, and assumed that a criminal and forensic investigation had commenced. The drive was imaged with EnCase and FTK.

The authors layered the foundation for determining and designing new applications for data acquisition from cloud computing environments [37]. The frameworks will be trustworthy and sound if the requirements of this research is fullfilled. Another contribution is the framework of this research will provide confidence and help forensic investigators, law enforcement and court to take decisions based on cloud data. The elastic nature of the cloud makes it possible for the criminal to commit crime and remove the data. Making the data tamper proof through acquiring provenance as part of the evidence will contribute to the strength of this research.

3.5.8 Digital Forensic Tools for the OpenStack Cloud

The design, implementation and evaluation of FROST are described in [38]. FROST is designed for OpenStack cloud computing, to enable forensic experts acquire evidence from the cloud controller. It was stated that provenance must be collected form the physical machines and not from the virtual machine instances. This will enable that the data will not be compromised by faulty and untrustworthy virtual machine instances. FROST collects data at the cloud provider, at the host operating system level underneath the guest virtual machines, and makes that data available within the management plane [38].

The management plane, exposed through website and application programming interface (API), is how users of OpenStack control the cloud, and where they start and stop virtual machines. The framework ensured that scalability, extensibility, compatibility and openness are preserved. When a user provisions a new virtual machine in OpenStack, a universally unique identifier (UUID) is assigned to the machine [38]. These UUIDs become children of the owner's root, and logs for that machine are appended as follows.

The subtree of any virtual machine has a depth of four for the year, month, and day of the log entry, with the log messages as leaves of the tree. Because the tree is constantly changing as new log entries are added, hash values for the intermediate hash tree nodes are re-calculated daily.

The mentioned structure enables a user to request any date range for any or all virtual machines, while reducing the additional overhead required [38]. Though the described mechanism, this research contributes by overcoming the trivial problem of remote evidence integrity by storing data in hash trees and returning evidence with cryptographic hashes. Under the existing mechanism, it is impossible to detect whether a malicious entity intentionally fails to log an event without access to the logger. Detecting provenance and chaining the existing provenance together with watermarking will ensure detection of the malicious entity.

3.6 Provenance Tracing: Watermarking Technique

Completion of linking cloud provenance information into a system provenance graph between different processes leads to the case of marking provenance graph for presence of malicious activities. Watermarking techniques have been used to mark the presence of nefarious activities and graph coloring techniques complements the identification procedure. This section discusses the current watermarking techniques used for security.

3.6.1 Secure Provenance Transmission

A novel technique for transferring provenance for streaming data addressing challenges like low bandwidth consumption, secure provenance transmission and high processing throughput [39]. Keeping track of provenance in such a highly dynamic environment is critical to ensuring data trustworthiness. Embedding provenance into the inter packet timing domain is the primary contribution [39].

The proposed solution is conceptualized as watermarking technique as the detected provenance is hidden inside another host-medium. Notable contributions include the embedding of provenance into inter-packet delays, thereby avoiding the degradation problem of watermarking techniques. However, the analysis of the received provenance was not highlighted by the authors.

3.6.2 Secure Resources and Data Coloring

A trust overlay network over multiple data centers was proposed for established trust between the service provider and data owners [40]. Watermarking and data coloring approaches were used to ensure integrity, confidentiality and availabiltiy of data. Since cloud computing uses shared files and memory locations, privacy and seurity can be compromised by adversaries. The watermarking technique consisted of coloring processes that uses data characteristics to generate colors such as data contents and expected value. Entropy and hyperentropy was used to add uncertainity.

The uncertainities were independent of the data contents [40]. Color matching process involves matching the expected value of a data to that of the data owner. If a match is found, the user is provided access to the data. The authors contributed to the field of watermarking using 2-way fuzzy login for data stored on the cloud. Hence the proposed watermarking scheme increase the trust and integrity of the cloud service provider. However, chaining of provenance and watermarking those to identify the attacker have been covered to a minimal extent.

3.6.3 Watermarking Method for Software Protection in the Cloud

Insider threat in the cloud is addressed using software watermarking technology in [41].Insider threats have been considered difficult to encounter in the cloud because the attacks are carried out by attackers inside the office. The insider threat attacks are often carried out by some of the most trusted employees in the enterprise. Thereby a framework to detect insider threats in the cloud need to be identified. The authors attached invisible labels to software copyrights using watermarking technique [41]. They aimed to embed watermarking using Ember function.

The Extract Function is used to retrieve the watermark from a watermarked program. No false positives were found in the extractor since those will always obtain a watermark [41]. The authors provided a mechanisms of mitigating the threat of unauthorized publication of software. They addressed the critical need of preventing unauthorized publication of stored software in the cloud. However, the authors did not address the important need of watermarking provenance information for detecting the presence of malicious entities in the cloud.

3.6.4 Data Provenance with Watermarks for Usage Control Monitors

Supplementing data monitors in distributed systems with watermarkers by tagging method has been described in [42]. In addition a mechanism to integrate provenance with existing data to prevent steganographic attacks was also proposed. Usage control monitors at the network level are expected to avoid data misuse through checking every data movement. The authors stated that such models were unfit for fast roll outs and low level attacks and steganographies. Existing controls monitors in [41], [43] were analyzed and it was found that all of those suffered from inaccuracies and misses when attacks are launched at fast speed since those do not take watermarking into account for increased security.

The authors proposed a watermarking technique that can enhance the capabilities of existing monitoring schemes. The watermarking approach would validate the legitimacy of the data packets by running usual policy checks such as matching creation date, owner and permissions. Next if required the system would extract all ids from a data packet's publicly readable watermarks. Afterwards the system would query watermark creators directly for provenance information [42].

The system will then re-evaluate and if needed pass the case to human auditors. Selected usage package monitors was surveyed to determine fault in cases of low level attacks and difficulty in backtracking. An algorithm was provided that integrates watermarking into existing usage monitors. However, no algorithm to attack provenance was provided. Watermarking provenance to ensure safety of data in distributed systems were not addressed. Chaining provenance information for backtracking was not considered. Finally usage of watermarked provenance to detect the presence of malware in distributed systems have been considered to a limited extent.

3.7 Privacy Aware Provenance

Recent technologies for provenance detection and scientific workflow systems are aimed for the group concerned with scientific workflow and provenance storage in database [44]. The objective of the research is to give an overview of the problem of managing provenance data for scientific workflows, illustrate some of the techniques that have been developed to address different aspects of the problem, and outline interesting directions for future work in the area. In particular, techniques for reducing provenance overload as well as making provenance information more fine-grained have been presented.

Provenance that go beyond the ability to reproduce and share results was examined, and workflow evolution provenance which can be leveraged to explain difference in data products was demonstrated, exploratory computational tasks were streamlined, and knowledge re-use was enabled. New applications that are enabled by provenance were also discussed, such as social data analysis [44], which has the potential to change the way people explore data and execute scientific experiments. On the basis of the mentioned experiments it was shown that workflow systems help scientists conceptualize and manage the analysis process, support scientists by allowing the creation and reuse of analysis tasks, aid in the discovery process by managing the data used and generated at each step, and systematically record provenance information for later use. Three key components of a provenance management solution are provided below.

1. ***Mechanism:*** The mechanism for capturing provenance
2. ***Data model:*** The data model for representing provenance information
3. ***Infrastructure:*** Infrastructure for storing, accessing, and querying provenance

Prospective and retrospective provenance capture, storage and retrieval of provenance data from database, modeling and representation of information are the primary purpose of the research in contention. In the context of scientific workflows, provenance is a record of the derivation of a set of results. There are two distinct forms of provenance prospective and retrospective as defined in the research. Prospective provenance captures the specification of a computational task that is a workflowit corresponds to the steps that need to be followed (or a recipe) to generate a data product or class of data products [44]. Retrospective provenance captures the steps that were executed as well as information about the execution environment used to derive a specific data product and a detailed log of the execution of a computational task.

In accordance to the two types of provenance mentioned above, a common feature across many of the approaches to querying those was that the solutions were closely tied to the storage models used. A wide variety of data models and storage systems had been used ranging from specialized Semantic Web languages and XML dialects that were stored as files and to tuples stored in relational database tables. Hence, they require users to write queries in languages like SQL, Prolog and SPARQL [44].

In addition to the above, use of provenance for social analysis of scientific data has been addressed in this research. Social websites and web based communities which facilitate collaboration and sharing between users, are becoming increasingly popular. Also a major challenge identified in the research involves provenance in education which stated that using a provenance enabled tool in class, an instructor can keep detailed record of all the steps which were tried while responding to students questions and after the class all those can be made available to students through the use of provenance.

The above research deals with the principal task of identifying the challenges and areas of application of provenance. The role and importance of provenance detection for successful administration of complex and distributed environments such as the cloud have been considered to a minimal extent.

Accountability as a Service has been identified for cloud computing environments in

[32]. The research presented a model in which cloud service providers are held responsible for accountability issues related to the services deployed in the cloud. In this paper, a novel design to achieve Trustworthy Service Oriented Architecture (TSOA) in the Cloud through enforcing strong accountability was proposed. In such system not only the root of a fault can always be concluded to the guilty participants, but also each conclusion is supported with non-disputable evidence [32].This is achieved by making the service provider accountable for the faults and breaches in the Service Level Agreement (SLA).

Similarly, the Service Oriented Architecture (SOA) as a design paradigm allows for new value added services in terms of compositions of existing services from different service providers [32]. The adoption of the cloud and SOA result in a cloud computing environment that enables highly dynamic and effective organizational collaborations. In such collaboration, each of the participants would behave according to the predefined and mutually agreed upon business logic and Service Level Agreement (SLA).

As stated above, any deviation from this agreed upon SLA is regarded as violations, and a robust mechanism is needed for its detection, logging and resolution [32]. The detection and prevention of failures under a composed service is complicated by the fact that composed services usually span several administrative domains, each of which will have its own interests and priorities. Building on the notions of trust presented, a trustworthy system is defined as a system that is already trusted, and continues to warrants that trust, because the systems behaviours can be validated in some convincing ways [32]. In such a system, the root of a failure or misbehaviour can always be identified and associated to responsible entity, and supported by non-disputable evidence.

3.8 Provenance based Identity Detection

Rule based provenance detection, tracing data provenance to ensure actual detection of identity has been discussed in alvaro2010boom. In the methodology, rule correlation engine was used which implements provenance algorithms on various existing cloud software, and performance in terms of data leakage were analyzed. The important of user-defined provenance and visual representation of provenance were highlighted in the research to a little extent.

With respect to the above issue, provenance detection for distributed computing has been analyzed. The research stated that the provenance detection scheme proposed in the research identified provenance data using Axis 2C, which is a service provided by Apache Axis and Mule that logs the actions to reproduce the results of the operations. Also the research provided standards that can be followed to detect and provenance data. However the methodology did not provide an infrastructure of cloud which can be used to detect provenance.

The importance of provenance was addressed and it was stated that provenance will play a central role in emerging advanced digital infrastructures. In the paper, the current state of provenance research and practice has been outlined, hard open research problems involving provenance semantics are identified, formal modelling, and security, and a vision for the future of provenance have been articulated [45]. The paper address the fact that technologies offer dramatic advantages, however they also exacerbate the hazards posed by buggy programs and dirty data. Digital information is easy to copy, change, and misinterpret. Current software systems do not provide the levels of repeatability, reliability, accountability and integrity achieved by the paper-based technology such as books, academic journals and laboratory notebooks that those are predicted to replace [45]. The solutions to the identified problems include provenance detection on the basis of semantics, traces, causality and identity, details of which are described below.

1. ***Semantics:*** Many forms of provenance could be captured for a particular system. At a bare minimum, where a particular piece of data comes from should be known [45]. While the above forms of provenance have already been introduced and studied in the literature for specific settings and languages such as relational databases, workflows, or file systems, there is a need for generally applicable, formal foundations for provenance. Moreover, since it can only be expected that provenance would become ubiquitously available if the effort of adding provenance support to systems is relatively low, an effective methodology that allows this general theory to be easily applied to new settings and languages should be developed [45].

2. ***Traces:*** Many forms of provenance are motivated by a desire to cache intermediate results and support efficient recompilation when the input changes. Incremental recompilation is a classical problem encountered in many different guises throughout computer science. Thus, when the input is changed, the effects could be propagated by replaying just a part of the trace. Such trace information ought to be computable from a sufficiently rich form of provenance [45].

3. ***Causality:*** These concepts are often invoked as motivations for provenance, but they are nontrivial. It is far from obvious how to make sense of informal claims that a given provenance record correctly captures a causal relationship, increases trust, or justifies a knowledge or belief, and most research papers dont even try. However, recent work on mathematical models of causality, trust, and knowledge may provide a good starting point for answering these questions [45].

3.9 Provenance Security and Integrity Obstacles

Provenance detection presents a number of challenges and obstacles for data forensic experts since the cloud is a distributed environment and there are relationships between multiple physical servers and between physical and virtual servers [46]. Hence detection of provenance for such an environment is a fairly difficult task that must be achieved. Ambrust et al have provided figures to quantify comparisons between cloud and conventional computing and identified the top six technical and nontechnical obstacles of cloud computing those are business continuity and service availability, data lock-in, data confidentiality and audibility, performance unpredictability, scalable storage, reputation fate sharing [46]. The identified obstacles are discussed to a greater detail in the following.

1. ***Business Continuity and Service Availability:*** Technical issues of availability aside, a cloud provider could suffer outages for nontechnical reasons, including going out of business or being the target of regulatory action [46].

2. ***Data Lock-In:*** Concern about the difficult of extracting data from the cloud is preventing some organizations from adopting Cloud Computing. Customer lock-in may be attractive to Cloud Computing providers

3. ***Data Confidentiality and Audibility:*** In special cases when the source of failure cannot be bound to a specific entity, procedures will be carried out to decide the violating entity in a best effort manner. The system shall suffer minimal delay due to dispute resolution [27].The security issues involved in protecting clouds from outside threats are similar to those already facing large datacentres, except that responsibility is divided among potentially many parties, including the cloud user, the cloud vendor, and any third party vendors whose value-added services have been bundled into the cloud offering [46].

4. ***Performance Unpredictability:*** The obstacle to attracting HPC is not the use of clusters; most parallel computing today is done in large clusters using the message-passing interface MPI. The problem is that many HPC applications need to ensure that all the threads of a program are running simultaneously and todays virtual machines and operating systems do not provide a programmer-visible way to ensure the above statement.

5. ***Scalable Storage:*** The opportunity, which is still an open research problem, is to create a storage system that would not only meet these needs but combine those with the cloud advantages of scaling arbitrarily up and down on-demand, as well as meeting programmer expectations in regard to resource management for scalability, data durability, and high availability.

6. ***Reputation Fate Sharing:*** One customers bad behaviour can affect the reputation of the cloud as a whole. For instance, blacklisting of EC2 IP addresses by spam prevention services may limit which applications can be effectively hosted.

To address the above issues, a methodology was adopted and it was shown that negative behaviors of attackers posing as customers can cause black listing of EC2 IP-addresses through span prevention which ultimately limits the application services of the cloud provider. Trusted email was suggested as a solution to the above problem, which leads to microcosm of the issue. However the use of log-based provenance detection to track the attackers has been discussed to a very little extent.

3.10 Issues in Provenance Research

The research is based on identifying provenance solutions for effective provenance detection and inference. However, the research focused to a little extent on the importance of providing a model on the basis of the solutions. If data forensics is to be successful, effective provenance detection models are required to identify disputed data in the cloud. Work on detecting provenance data for investigation of malicious activities on stored files and databases have been conducted to a limited extent. Secure provenance which implies tracking and recording of ownership and process history of data, are essential for data forensics. A scheme that keeps the provenance data confidential from general users in case of sensitive records should be provided, which will ensure successful maintenance of metadata to aid data forensics.

Provenance is an unexplored area in cloud computing, where a number of security challenges exist. The provenance data should be stored in a way so that original provenance information should not be forged by malicious users. The model of provenance design should be such that the provenance data can be revealed only by a trusted authority, and no one else. Provenance data may be platform specific or may not be reliant on a given platform. Developing provenance schemes to ensure interoperability between different types of data is an area of keen research interest. If platform independency and interoperability of metadata is achieved, the migration of provenance data from one system to another can be achieved.

Obtaining and representing provenance metadata without violating security or privacy of the users is an active research issue. In addition to these attributes, developing a model that provides unforgetability and interoperability is a highly sought-after architecture. A design with the mentioned capabilities will achieve the desired attribute of provenance data sustenance even when the artifacts are removed.

Although extended studies have been carried out in provenance, the main challenges and requirements to implement cloud provenance is an active area of research. Considering the dynamic and complex nature of cloud computing, a model that would link log and audit data collected from multiple infrastructures is a challenging task. These challenges should be identified to a greater detail in order to ensure that they can be mitigated for reliability of cloud computing.

Cloud provenance collection must be real time as the data is accessed and manipulated. Time tagged provenance detection is an issue, however ensuring time synchronization between the different physical machines like cloud and node controllers is another challenge.

Provision of a model for an API for collection of provenance data from the users and making that data available on the web for easy access is also a vital research issue. The most recent research objective is to incorporate the users with the obtainment of provenance or attribution data. This will also enable the system administrator to check whether the user comply with the provided information. Hence this model is highly demanded.

A well designed user APU-GUI is needed to permit user annotation and application specific provenance. As a result, challenges of automated provenance collection can be mitigated and at the same time user involvement and increased semantic knowledge on provenance data can be ensured. If the data was generated from non-provenanced sources, a good solution for incomplete provenance data is to allow signed user input, with the help of a well designed API. Assurance that provenance data is not forged by an adversary and mechanism to detect and prohibit suspicious user annotation is still an open research question. The provenance model should be such that an administrator should be able to monitor provenance data. The model should also ensure that provenance data is encrypted and should have access control policies like the one described above.

A comprehensive provenance framework is essential for researchers to verify quality of data, reproduce scientific results in peer-reviewed literature and validate scientific processes [47]. This framework is required to ensure the pre-publication and post-publication states of data in translational research activities.

Provenance data without incorporation to real life web data is of little use. Approaches need to be defined for incorporating provenance data to the data on the World Wide Web.

This is needed for quality assessment of data on the web since web data undergoes a large amount of replication, query processing, modification and merging [47].

The provenance data that are not very sensitive and needed by the users to verify the trustworthiness of data on the web should be made available to the users. Also, the provenance metadata should be linked to the data which it describes. To make the provenance data part of the web data, both must adhere to the same publication principles [47]. A method that automatically synchronizes provenance data with actual data should be provided. The goal of this framework is to ensure the quality and trustworthiness of provenance data.

In case of cloud computing, lack of physical access presents a new challenge for provenance collection. Due to decentralized nature of cloud computing, traditional methods of collection metadata about data is no longer a practical approach [47]. Research needs to be done whether it is possible for cloud customers to carry out provenance detection on their data which is stored on the cloud from a technical standpoint. Methodologies to perform such investigations needs to be provided as it will give more control to the customers of cloud computing.

Similarity and overlapping between trusted computing and cloud computing provenance should be identified. Architectural idea for a trusted provenance system and the requirements for effective integration of trusted computing systems and provenance systems should be identified for effective provenance detection.

Mechanisms to develop provenance storage in the cloud are an area of significant research interest. Cloud storage is affordable and scalable. So if an effective system for storing provenance data on cloud storage can be engineered, the issue of data storage cost and continuous growth in volume of attribution data can be mitigated.

The methodology of using cloud as a standalone system to store provenance data is also an important issue of research. The issue is justified since it tries to propose a model of storing provenance data using a standalone cloud storage system, thus providing a new way of database-cloud storage co-ordination.

The target of the application is to identify the granularity for provenance data, and group those on the basis of the granularity. Despite the extensive research interest shown in the field of cloud computing provenance, a number of drawbacks exist. The provenance research works have only focused on database systems, operating systems and automated provenance detection. Research on how cloud can replace existing methods like databases systems to store provenance data has been carried out to a limited extent.

Community standards such as the Open Provenance Model allow uniform interpretation and exchange of metadata, but do not prescribe query specifications to retrieve provenance. A system may perform a pipeline of tasks that will execute in a specific environment due to platform dependency; however, little work has been done to devise a methodology for storing this data in a single repository and in real time as the pipeline of tasks is carried out. There is scope for research in the mapping between the physical machines and virtual machines in cloud computing. Since in cloud computing, most of the applications are ran on the virtual machines, the provenance data must be recorded even when the virtual machines are terminated. However, very little research has been done in this regard. Data reuse and transfer across multiple platforms must be accompanied by provenance data, otherwise tracking this data and ensuring the trustworthiness of data will become debatable. The research papers under consideration consider this requirement to a very little extent.

3.11 Summary

Digital forensic experts can use provenance information obtained from compromised systems to identify the root cause of the malicious act. For a dynamic and distributed system such as the cloud, provenance plays a significant role in forensic investigation because of the following attributes:-

1. It aids to identify the source of the unwanted activity.
2. Effective interpretation of the malicious behavior and can provide useful knowledge on the nefarious activities through monitoring the processes executed in the cloud.
3. Provenance from a large collection of virtual and physical machines can be collected.
4. Establishing interrelationship between the large provenance body can provide state of the art information on machine behavior.

In this chapter the related research in the field of distributed systems provenance and securing those with effective analysis models have been highlighted. The motivation of the research in securing provenance data and effective representation of provenance analytic for ensuring integrity of the cloud have been specified. The significant research contributions have been identified in this chapter and the unsolved problems of securing and presenting provenance for analysis have been highlighted in different sections of this chapter.

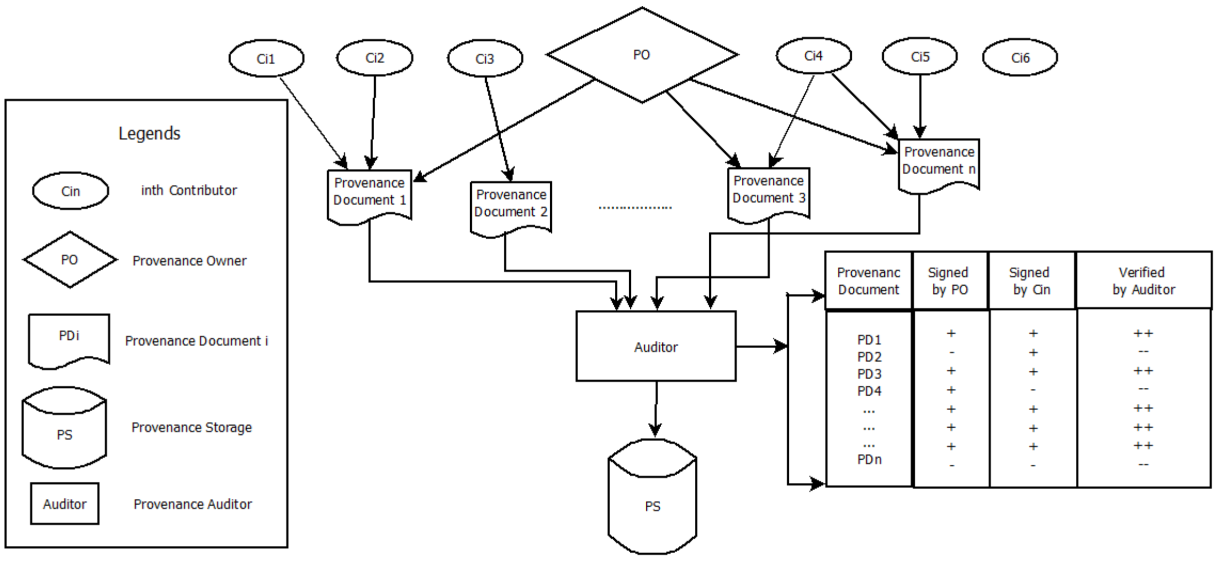
Chapter 4

Securing Provenance

The statements of the previous chapter emphasized the importance of not only capturing provenance, but also making the provenance tamper-proof to prevent unauthorized manipulations. The framework proposed here consists of encapsulation of the captured provenance meta-data with the original data using the proposed *ProvCapsule* algorithm. Next *ProvOCal* and *ObProv* algorithms are used to ensure that the bounded provenance are encrypted with private keys from Provenance Owner (PO) and Auditor. Both PO and Auditor processes are under the control of two different trusted bodies to ensure accountability and prevent the unwanted practice of single point evaluation. The PO and Auditor processes share the public keys with each other.

After encryption the provenance information must be stored at the secured end of the trusted Auditor. Algorithms and framework of the trusted model discussed above are explained in this chapter. At the same time experiments to test the effectiveness of the proposed provenance securing algorithms are conducted with real life cloud environment using OpenStack Essex. Scenarios depicting system processes and socket triggers in the cloud are highlighted and provenance is captured for those,encapsulated and encrypted using mechanisms proposed in those chapter. Finally Auditor algorithm is implemented at the end of the trusted body to ensure provenance persistency and achieve data integrity.

Experiments are conducted for files at the application layer in the cloud. Users may copy, request or upload files in the cloud environment to ensure persistent storage. Provenance information are collected from those using the proposed ProvCapsule, ProvOCal, ObProv and Auditor algorithms.



Provenanc Document	Signed by PO	Signed by Cin	Verified by Auditor
PD1	+	+	++
PD2	-	+	--
PD3	+	+	++
PD4	+	-	--
...	+	+	++
...	+	+	++
...	+	+	++
PDn	-	-	--

Figure 4.1: Architecture of the proposed provenance security framework

4.1 Proposed framework for provenance security

The proposed framework consists of two trusted bodies that compose of many processes at the application level. The framework for secured provenance cognition is stated in Figure 4.2. Data requested in the cloud are secured jointly by the Provenance Owner and Auditor processes stated in the legend of Figure 4.2. The critical processes identified in the legend are described in the following.

1. Provenance Owner (PO): The entity called PO is responsible for providing the required encryption key for securing the provenance information from the side of the cloud service provider. PO generates public and private keys and shares its public key with the Cin.

2. Contributor ith (Ci): A contributor is the vm-instance provided to the customers that executes the application programs at the application layer.

3. Provenance Document (PD): The document or meta-file that stores provenance from the data files that are accessed in the cloud. The provenance document is the file

that is signed and encrypted by the keys of the PO and Auditor. The PO is under the control of the cloud service provider and hence its private keys are only known to them.

4. Auditor: The process that verifies that the provenance information is coming from an eligible entity through decryption of the message and comparing the data with provenance information such as time-stamp, instance id, date and request type. The responsibility of the auditor is to verify every single PD that reaches it. The auditor is under the control of the forensic experts and the private keys are unknown to the cloud service provider. Since the *Provenance Owner* (*PO*) and the *Auditor* are under two different groups, integrity will be ensured and the probability of forgery through nefarious activities is reduced manifold.

5. Provenance Storage (PS): The storage is responsible for storing the obtained provenance file in its encrypted format. The provenance file must be encrypted by the PO and Auditor before it can be stored at the PS. For effective query processing and optimization in terms of overhead for data retrieval, all the information is stored in unstructured format and retrieved using Not-Only SQL (NoSQL).

4.2 Difference with existing research

The proposed framework emphasizes on the security of provenance information. Which is historical derivation meta-data about the original data. However, existing research methodologies enforce on encryption of the critical data instead of provenance information [4], [6]. At the same time, most of the research on data security emphasize encryption in the onion structure mainly because of its simplicity of data retrieval [17]. However, security of a provenance information of a distributed system such as the cloud cannot be ensured only

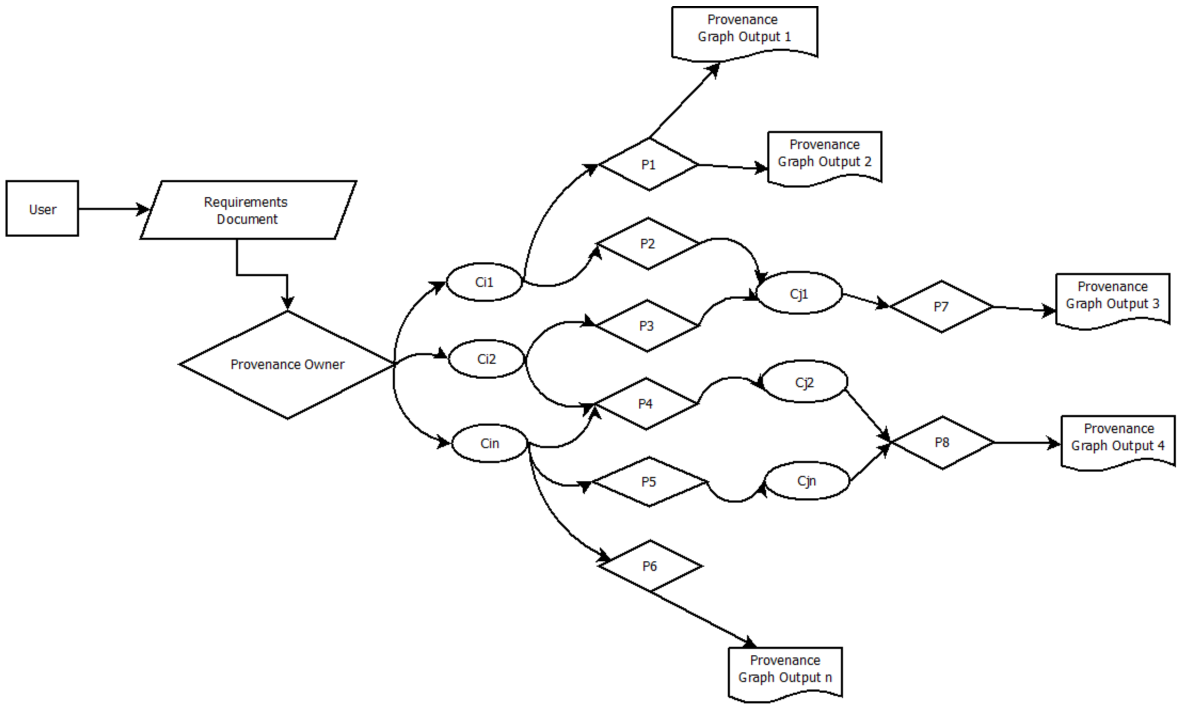

Provenance Graph Output 1
Provenance Graph Output 2
User
Requirements Document
P1
P2
Ci1
Cj1
P7
Provenance Graph Output 3
P3
Provenance Owner
Ci2
Cj2
P4
P8
Provenance Graph Output 4
Cin
P5
Cjn
P6
Provenance Graph Output n

Figure 4.2: Encryption mechanism of the proposed framework

by onion type encryption. Causality needs to be ensured to establish and verify the linkage between physical and large number of corresponding vm-instances.

Next, the proposed methodology identifies two independent processes, called Po and Auditor, responsible for ensuring the security and anti-tamper-proneness of the provenance information [24], [25]. The control of the *Auditor* and *PO* are in the hands of the forensic expert and cloud service provider respectively. As a result there are two independent signing and verifying bodies that ensure total transparency and integrity. However existing research focuses on implementing both the processes under a single control, thus making the system prone to tampering due to single point of control.

Another fundamental difference with existing research is that the procedures used for signing and verifying provenance information are time consuming since the authorization process is manual [24]. In case of protecting the system from malware, such encryption and verification techniques are programs that can be affected and tampered through the manipulation of the certifying algorithms. However, active-threading technique is proposed here for binding provenance with the original data file in real time, thus the process of forging provenance ownership is prevented in the proposed framework through real-time binding of provenance meta-data to original data. This methodology also reduces time and low-overhead since only critical information like time-stamp, date, target vm and source vm are contained in the provenance file unlike detection of detailed meta-data.

Figure 4.3 identifies the flow of sequence to ensure provenance security. The user can access a cloud application through a browser or may request some data file in the cloud storage. The data is stored in the C_i and provenance is collected from those using the ProvCapsule algorithm. Next the provenance file containing the time (t), source vm-id (svm), target vm-id (tvm) and date are stored in the provenance file which is then encrypted by PO using its private key. PO generates a private/public key pair and shares it with Auditor. Auditor receives key $K1$ from PO and provides its own public key $K2$ to PO.

The second level of encryption is carried out by the auditor and the provenance file is sent to the signature process to be verified, if the time-stamp, source vm, target vm and date matches, it will be forwarded to the PS, otherwise it will be discarded and the PO and Auditor will both be notified about the issue.

4.3 Security Goals and Assumptions

The research goal is to prevent the unauthorized or malicious access and nefarious manipulation of provenance information in the cloud. The main research question addressed in this section is given below. Is the cloud provenance data being tampered by a malicious entity?

The assumptions for the proposed algorithms include that the cloud environment is in Linux platform and the cryptographic algorithms are implemented properly. The forensic

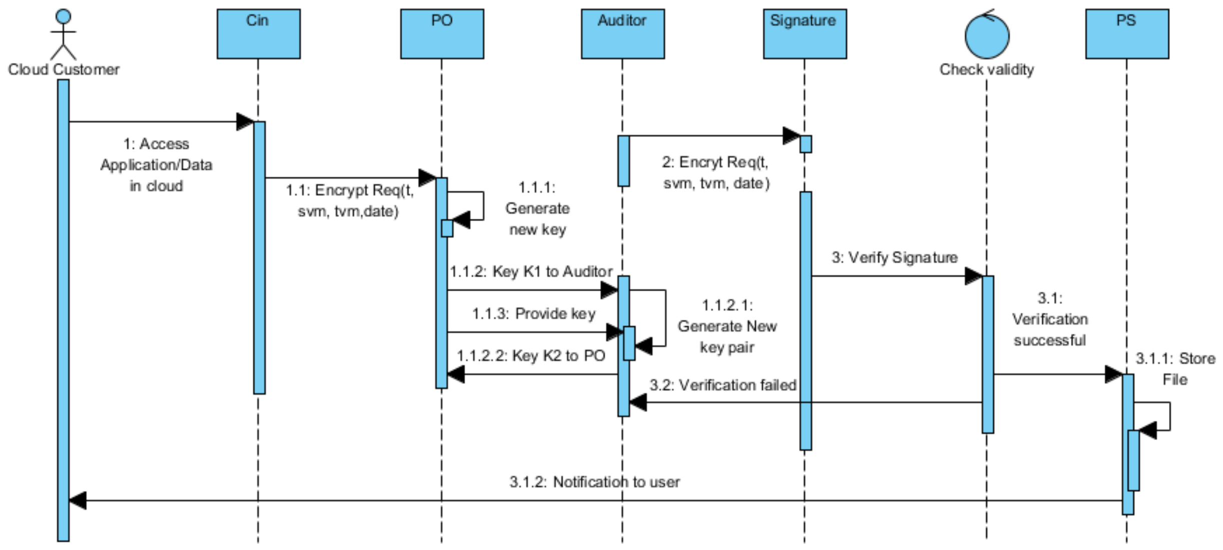


Figure 4.3: Sequence of actions and processes to secure provenance information

investigators who own and maintain the auditor process is trustworthy. Also, it is assumed that program scripts running at the kernel level are not tampered by the cloud administrator.

4.4 Data Confidentiality and Audibility

Security of the cloud is often the most cited objection since customers are severely concerned about the security of the data they store on the cloud. The security issues which are used to protect the cloud from attacks are similar to the ones implemented at large scale data centres, with the exception that both customers and not only cloud administrators are responsible for the security. In addition to the two parties, a third party may be involved for the security of the cloud as well which includes providers of value added services which are incorporated into the cloud. For example, RightScale provides value added services, a few of which are automatically incorporated with Amazon Elastic Cloud Compute (EC2) and helps dynamic scale up or scale down of EC2.

The cloud users are responsible for the security of the vm instances which are sanctioned. The cloud administrators are responsible for the security of the physical layers, as well as the security of vm layer by monitoring data regarding launching of vm instances and tracking logs of file access and changes made to those. So security of the intermediate layers and vm instances are shared among the cloud administrators and users. If the users are exposed to greater details of vm instances, more responsibilities are handed over from the administrator to the user. Some cloud providers outsource the security maintenance task to third parties who have independent IT management to manage the cloud services. Amazon EC2 users have more technical responsibility than Azure users, who in turn have more technical responsibilities than AppEngine users [11].

The primary security mechanism that can be used in todays cloud infrastructure is log based provenance detection. Log based provenance detection can be used to monitor the vm instances from launching to termination [11]. This is a powerful tool to ensure confidentiality and audibility, since attempts by attackers to deactivate vm instances or stop underlying cloud processes can be identified. However, provenance detection must be based on log analysis and it is not bug free as certain safe actions can be considered as potential attacks. Incorrect provenance detection mechanisms can result in attacker having unauthorized access to the customers vm state information. The challenges are similar to large data centres which do not implement cloud computing, where various programs need to be protected from each other physically. Any large data centre, be it cloud or non-cloud, will look to detect provenance of the vm layer to detect malicious activities.

Similarly, audibility must be added as a security mechanism which would include provenance detection of physical data centres, which is beyond the reach of any outsider, be it a user or a malicious attacker. Provenance detection can provide full security to the vm layers and also to the physical layers, and makes the cloud arguably more secured as compared to other vendor-provided security applications which act only at the vm layer. Provenance detection scheme can ensure detection of any unauthorized activity about vm instances as well as the system itself. Such new features reinforce the shift in viewpoint of cloud computing from only the physical layer to the vm layer as well.

4.5 ProvCapsule algorithm

Initially, each file contains a provenance metafile called F_{PROV} identified by a unique id B_{ID}. Original files identified by F_{ORG} are stored in $DataFile[x]$ array. F_{PROV} are stored in $ProvFile[y]$ array. Each F_{PROV} encapsulates specific F_{ORG} together with its provenance information. Active-Threading is ensured as the algorithms read each F_{ORG} and treat those as individual objects. Next the methodology sets B_{ID} for each F_{ORG} and F_{PROV} to identify those uniquely.

Algorithm 1 Provenance Capsule at Cloud Activity Layer

```
procedure ProvCapsule($a, b$)
    $B_{ID} \leftarrow key$ and $F_{ORG} \leftarrow DataFile[x]$ and
    $F_{PROV} \leftarrow ProvFile[y]$ and
    $P_{SB} \leftarrow Loc[z]$
    while $B_{ID} \neq 0$ do
        Read inputs in $F_{ORG}$
        Record for every object $a$ in $DataFile[x]$
        while $a = DataFile[x]$ do
            $B_{ID} = a.B_{ID}$ and
            $F_{ORG}$=$F_{ORG+1}$
            $DataFile[x+1]$=$F_{ORG}$
            continue
        end while
        Record for every object $c$ in $ProvFile[y]$
        If $B_{ID}.F_{PROV}$=$c.F_{PROV}$
        $B_{ID}.F_{PROV} \leftarrow B_{ID}.F_{ORG}$
        $B_{ID} \leftarrow B_{ID} + 1$
    end while
    $ProvFile[y]$=($Loc[P_{SB}]$,$P_{SB}.Exec$,$B_{ID}$)
    return $B_{ID}.F_{PROV}$ & $length.DataFile[x]$
end procedure
```

Once the B_{ID} of an object currently in queue matches an object stored in the $DataFile[x]$

array, it will read the input information I_a that is a subset of the original set of inputs I. This process is used to detect and associate provenance information such as date, time, operator and operation of specific $DataFile[i]$ into the $ProvFile[j]$. The data are stored in a separate F_{PROV} file which is then bounded to the original file as a metafile. The F_{ORG} is then transmitted in the cloud due to commands of the activity layer.

4.6 ProvOCal algorithm

The algorithm for mapping provenance for memory reads and disk writes are discussed here. Two variables g and h are used to compare length of memory read and length of disk writes respectively. After those are recorded, g and h compared with the obtained results. $MInf$ is the variable for storing memory information of F_{ORG} and recording its memory length. *MSum* adds the length of disjoint specific blocks to calculate the total size of memory required. The result is added to the minimum between one less than the total length or g.

The calculation of disk location and length of the $DataFile[x]$ and $ProvFile[y]$ is similar to the $MemInf$ calculation, with *SBSum* saving the final value. Finally the difference in the length of *MSum* and *SBSum* are calculated on the basis of the matched B_{ID} to determine whether the file is loaded into memory from disk in a compressed format. Hence effective information from the activity layers in terms of memory and disk information can be identified from the algorithm. Next the value of h is incremented as the algorithm returns the disk and memory consumption of the F_{ORG} object at the activity layer of memory management.

Algorithm 2 ProvOCal Algorithm for Provenance Detection in Memory and Disks

```
1: procedure ProvOCal($a, b$)
2:     $g \leftarrow 0$ and $h \leftarrow 0$
3:     $MInf = Loc[P_{SB} + 1]$
4:     while $g < len.MInf$ do
5:         $MSum = MInf$
6:         $S = min(g, len.MInf - 1)$
7:         $MSum = MSum + S$
8:     end while
9:     while $h < len.SBInfo$ do
10:        $SBSum = SBInf$
11:        $T = min(h, len.SBInf - 1)$
12:        $SBSum = SBSum + T$
13:    end while
14:    If $len.B_{ID} < len.MemInf$, then
15:    $f \leftarrow len.MemInf - len.B_{ID}$
16:    h = h+1;
17:    return Memory and Disk block consumption for provenance capsule
18: end procedure
```

4.7 ObProv algorithm

The *ObProv* algorithm is aimed to encapsulate the provenance information in a specific message body. Hence provenance information produced by *ObProv* contains only the information need by the auditors or digital forensic experts. The M_{sg} is a functions that will encapsulate the captured provenance information of ProvOCal into a specific provenance message. At the same time users are allowed to specify what type of provenance information they want to obtain and encapsulate. The M_{bd} consists of the message body. The variable g keeps the length of the size of the provenance information and messages produced by the algorithms, and also acts like a counter.

The array $A[x]$ consists of the messages that are stored as strings together with the timestamp T. The timestamps are used to compare the authenticity of the requests in the later algorithm. The request body $Req(T, T_{vm}, S_{vm}, Date)$ ensures that the critical provenance information as specified by the digital forensic experts. The contents are $T, T_{vm}, S_{vm}, Date$ to

Algorithm 3 ObProv algorithm for capturing provenance at the application level

```
procedure ObProv($M_{sg}$, $M_{bd}$, $g$)
    $M_{pr} \leftarrow Req(T, T_{vm}, S_{vm}, Date)$
    $M_{bd} \leftarrow msg_1, msg_2, .., msg_n$
    $Msg \leftarrow M_{pr} + M_{bd}$
    $A[x] \leftarrow 0$
    $g \leftarrow 0$
    while $g <= sizeof(Msg)$ do
        $A[g] = Msg$
        $g = g + 1$
        $ObProv \leftarrow ProvOwnerc, d$
    end while
    If $len.Msg < len.M_{pr}$, then
    $Msg \leftarrow len.Req(T, T_{vm}, S_{vm}, Date)$
    g = g+1;
    return Provenance information file $Msg$
end procedure
```

represent time, vm-id, storage allocated for the vm and date of access respectively. If the counter of g that consists of the size of the message that does not exceed, then the message Msg is placed the the array $A[g]$ in the location identified by the current value of g. At the same time *ProvOwner* is read from *ProbCapsule* and placed in the owner variable of *ObProv*. Finally the *Msg* is assigned the *Req* body and send to the next algorithm called *Auditor*.

The algorithms discussed above function in line to detect system level provenance, at the same time those are synchronized to ensure proper activity from provenance cognition, to secured provenance encryption. The next algorithm verified Hence there is a sequence of 4 algorithms that work closely to achieve secured provenance of the web cornerstone data stored in the cloud.

4.8 Auditor algorithm

The auditor of the process is responsible for verifying the request on the basis of the timestamps. The variable R_1 and R_2 are used to measure the size of the buffer array and at the same time implement the check constraint that it is under the limit of the initial consideration. Both variables are initialized to 0 to ensure that it is not reduced variables are incremented and counted, hence providing a variable that can be used as a counter function.

Algorithm 4 Auditor: Setting permission for provenance access through timestamp authentication

```
1: procedure AUDITOR($R_1$,$R_2$,$R_3$,$R_4$,$ObProv$,$size$)
2:     $R_1 \leftarrow 0$
3:     $R_2 \leftarrow 0$
4:     $Per_i \leftarrow 0$
5:     $size \leftarrow 1000$
6:     $Buf[size] \leftarrow ObProv$
7:     while $R_1 <= sizeof(Buf[size])$ do
8:         while $R_2 <= sizeof(Buf[size])$ do
9:             $R_1 = sizeof(Buf[size] + Msg)$
10:            $R_2 \leftarrow En.(Req(T, T_{vm}, S_{vm}, Date))$
11:            $R_1 = R_1 + 1$
12:        end while
13:    end while
14:    while $R_3 <= sizeof(Buffer[size])$ do
15:        while $R_4 <= sizeof(Buffer[size])$ do
16:            $R_3 \leftarrow timetoread.Buffer[size]$
17:            $R_4 \leftarrow timeofarrival.Buffer[size]$
18:            If $R_1 = R_3$ and
19:            If $R_2 != R_4$
20:            $Per_i = 1$
21:            else $Per_i$=0
22:        end while
23:    end while
24: end procedure
```

The two other variables R_3 and R_4 that are used to store the timestamp when the first message is read and the arrival time of the second message respectively. The times of arrival of the new request is matched with the time when the first request if made. At the same time the vm-instance id that is making the request is also recorded.

The two timestamps of the requests do not meet even in the case that the vm-id of those matches, the Per_i will be set to 0 and the permission to access the provenance information will not be allowed because the mismatch of timestamp is caused by a different vm-id philishing the identity of the authentic vm. The difference in the timestamp will ensure that the vm requesting access does not have the authorization and it is philishing identity.

4.9 Transition of proposed architecture

The four algorithms discussed above need to communicate with each other using message passing protocols to ensure that data is collected by one algorithm, sent for encryption to the next and finally validated by the remaining algorithms for digital investigation. The proposed algorithmic architecture consists of a number of states that must be transited. At each state proper verification mechanisms need to be implemented to ensure that the process satisfies the requirements needed to advance to the next phase.

The start state in Figure 4.4 consists of provenance information being gathered in the request body $Req(a,b,c,d)$ and sent to *ObProv*. Next the request is sent to *ProvCapsule* so that the collected provenance are encapsulated to its original data file to sid in forensic analysis. Next the bounded provenance and data files are sent to the encryption scheme to obtain the data in encrypted format. During encryption the provenance owner and auditor provide public keys and encrypt the information with respective private keys. The hash algorithm is implemented to ensure the integrity of provenance information. Hashed chain format of encryption as opposed to onion type encryption is used to accommodate the large volume of provenance information as generated by cloud processes and applications.

After encryption, the provenance information is sent to the provenance storage that is maintained in the hands of the auditor since it is assumed that both consumers and service

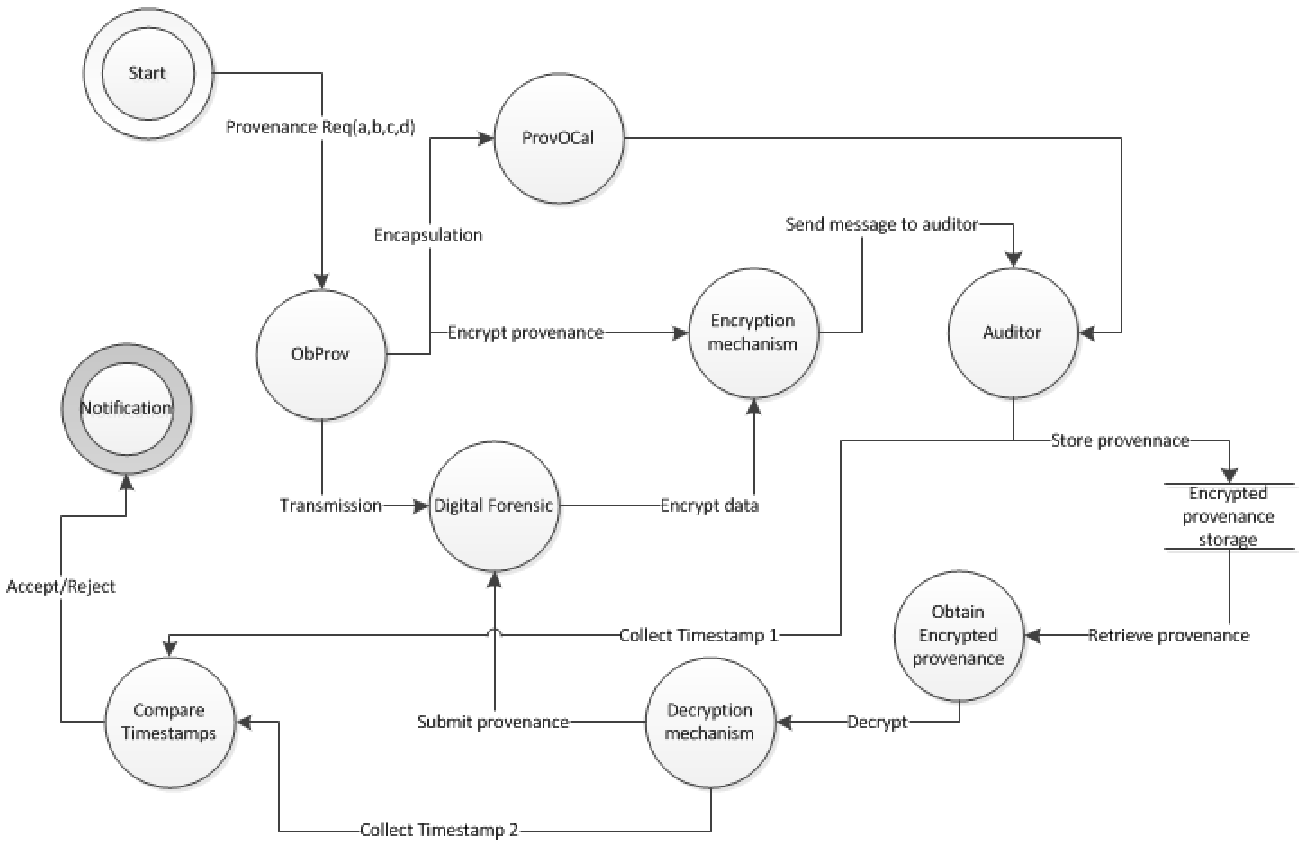


Figure 4.4: Sequence of transitions from one state to another in the proposed architecture

providers consider the auditor to be a trusted body. The provenance is stored in the storage for a specific time period before archiving. During the course of digital investigation if provenance information is required, it is retrieved from the storage using the specific id assigned to those by the *ProvCapsule* algorithm. However, the obtained provenance information is still in encrypted form, hence it is decrypted using keys from the *PO* and *Auditor* processes for analysis. Next the auditor uses the provenance information for digital forensic evidence and analyzes those for effective determination of the nefarious actors and activities.

4.10 Summary

The chapter aimed to ascertain a solution to the bottleneck of capturing provenance data in a widely decentralized system such as cloud computing. Algorithms for detecting provenance and encapsulating those on the specific objects have been proposed in this chpter. Compared to traditional provenance detection techniques which function in standalone systems, the proposed active-threaded algorithms are capable of detecting provenance in a virtualized environment.

As stated earlier, the proposed mechanism detects provenance for enabling forensic experts to use it in digital forensic investigation. Ensuring performance of the algorithms for web cornerstone data provenance analytic algorithms in the cloud should be conducted.

Chapter 5

Using Web Cornerstone Data with Provenance for Effective Security Intelligence

The cornerstone data are widely available on the web and specific cornerstone data are used to mitigate specific user needs. Based on the requirements of accuracy, the importance of melding keystone an cornerstone data is manifold. There is an increasing research need to amalgamate cloud keystone provenance and cornerstone data to achieve high accuracy Cloud Security Intelligence (CSI) in a short length of time. Hence the following research question is addressed in this chapter.

How can structured keystone data be melded with unstructured cornerstone data to synthesize effective CSI with increased accuracy?

Attributes of structured keystone data and situational, unstructured cornerstone data have been identified in this chapter for differentiating the two types of data. Aspects of volatility, longivity, availability, host, source, etc have been taken under consideration for the differentiation. Next the features of cornerstone data that have been taken as validation criteria to judge the integrity of the websites have been identified such namely freshness, user ratings, owner rating and watchword matching. The two sources of cornerstone data have been defined as *Valid Cornerstone* Data and *Volatile Blog and Social Cornerstone* Data. The proposed methodology defines the transition and merging of the structured provenance keystone and unstructured situation cornerstone data.

Algorithms for parsing cornerstone data is provided in this chapter. Secondly effective algorithms to conduct keyword based structuring of the cornerstone data is proposed. Thirdly algorithm for determination of the reliability and integrity of the cornerstone data is proposed based on owner ratings, user rating and freshness of information. Algorithm *Cornerstone* is responsible for identifying the valid sites on the basis of query formation and user feedback. The *CrawlCornetstone* algorithm goes about the website and collects information from it in unstructured format. Finally the *AssignValidityValue* algorithm is responsible for effective rating of the web pages that ensures the importance of data in decision making procedure when the web cornerstone data are involved. The mathematical evaluation provided in this chapter is critical to the evaluation of the effectiveness of the proposed algorithms since those reflect the merging procedure of structured and unstructured data.

5.1 Merging Provenance with Web Cornerstone Data for Improved Analytics

As stated in the previous chapter, securing provenance after detection have been addressed as the research issue. The results of the earlier experiments gives rise to another issue of analysing provenance data through fusion with upto date unstructured data obtained through the World Wide Web for accurate Security Intelligence (SI). The integrity of the web data must be ensured, and due to its unstructured format and large volume, it is often referred to as Web Cornerstone Data. Here the structured provenance data obtained from the system are referred to as keystone data and the unstructured data obtained in large volumes from the web are referred to as cornerstone data.

Even if the keystone provenance data is secured by the mechanisms stated in the previous chapter, rapid decisions have to be made in times of cloud security threats based on reliable data, and evaluations of potential consequences have to be made. Hence, the necessity of fusing keystone with cornerstone data will enrich the knowledge base of the Digital Forensic Investigator manifold and increase the certainty of achieving accurate SI.

Previously, secured storage of provenance in multi-dimensional Data Warehouses have been discussed in [1]. Those data can be queried using Online Analytical Processing

(OLAP) strategies to reduce overhead in terms of query execution time. Here the use of OLAP to query Keystone provenance is proposed and the performance compared with traditional query processing. OLAP will enable interactive queries to be executed and upon merging those with Web Cornerstone data, effective decisions can be taken by the Digital Forensic Investigator for SI.

Static multi-dimensional provenance is identified as keystone data due to its tamper-proofness and reliability achieved through the solution of the previous chapter. This chapter target to fuse keystone data with volatile unstructured data called cornerstone. Cornerstone data are obtained from the web, and the name is given mainly due to the criticality, unstructured format and volatility of those. The aim of this chapter is to integrate keystone and cornerstone data to get more accurate predictions on Cloud Security Intelligence (CSI).

5.2 Keystone and Cornerstone data

Well informed and effective decisions often require tight relationship to be established between keystone data and cornerstone data that are outside the scope of the auditor [1], [2]. The cornerstone data will be in relation to operating system specific nefarious entities, process specific malware or file-corruption based processes. Following definition is applicable for cornerstone data:

Cornerstone data are those that are needed fr security decisions but do not persist as part of keystone provenance. Cornerstone data are available on WWW and have a relatively narrow focus on a specific Cloud SI issue and the lifetime of those is relatively short compared to keystone.

The assumption for the above scenario is that cornerstone data are randomly scattered across heterogeneous and unstructured sources available on the web. Significant attribute of the cornerstone data is that those tend to be highly dynamic in comparison with keystone

Table 5.1: Attributes of Cornerstone and Keystone data

Attributes	**Unstructured Data**	**Structure Data**
Reliability	highly volatile	reliable to significant accuracy
Host	web hosting services	Data Warehouses of the Auditors
Source	web pages	database of provenance owners
Availability	not guaranteed	all time
Longivity	short	relatively long
Analysis	runtime	design time

data. Provenance based keystone data are used to address a large set of decisions and problems. Table 1 shows the difference between cornerstone data and keystone data.

5.3 Difference with existing research

In [2] the authors tried to understand the characteristics of workloads running in MapReduce environments through provenance analysis. It was addressed that understanding the characteristics of workloads for different scientific applications running on the cloud will be beneficial for both the cloud service provider and cloud customer. The cloud service providers can use this information to better analyze specific workloads and make effective resource provisions for those. The customers can comprehend the factors that affect the performance. Also users can analyze the cloud resources that are important for those. For experimental purpose, ten months of provenance trace data from M45 server in Yahoo was analyzed [3]. The system runs Hadoop and provides free cloud resource for system research at selected universities. The M45 cluster consists of approximately 400 nodes, 4000 processors, 3 terabytes of memory, and 1.5 petabytes of disk space. The clusters runs Hadoop and provisions small virtual clusters over a large physical cluster [3].

Statistical analysis of the server performance based on provenance data was provided. In addition, analysis of provenance logs identified the performance and failure characteristics of Hadoop jobs running in large scale real life clusters. The identification of security vulnerabilities and malicious activities was not done by the provenance detection scheme. At the same time the provenance results did not yield any performance degradation caused by malware actions.

Analyze logs of electronic mail to identify information about Map and Reduce (MR) programs at the MR level abstraction of Hadoop was proposed in [4]. The proposed algorithm helps to understand and analyze Hadoop jobs in real life clusters. In addition performance benchmarks and overheads for different tasks in the cluster have been evaluated. The proposed algorithm extracts local node centric Hadoop provenance data. Next the obtained data are correlated among different nodes and processed to obtain performance information of job structures in Hadoop [5]. The analysis result yields a unique end to end representation of causal activities in the cluster also known as Job-Centric Data-Flow (JCDF). State machine abstractions for Hadoop execution was used in the methodology. However, such mechanism cannot be used to detect vulnerabilities like SlowLoris and Code Injection Attacks in the cloud.

5.4 Proposed methodology

The proposed algorithms will conduct search to identify unstructured cornerstone information from the web. The identified sources will be based on one of the following criteria.

1. ***Watchword frequency:*** Standard criteria from information retrieval could be used here, taking both data and their metadata, e.g., tags, into account.

2. ***Owner rating based:*** This involves the rating provided by owners of the web sites compared together with the significance of the owner itself. For example, the owners of Symantec and Kaspersky have high rating in accordance to the algorithms. Hence those sites rated by such popular owners will be significantly improtant for decision making processes of the proposed mechanism. On the other hand, the ratings of

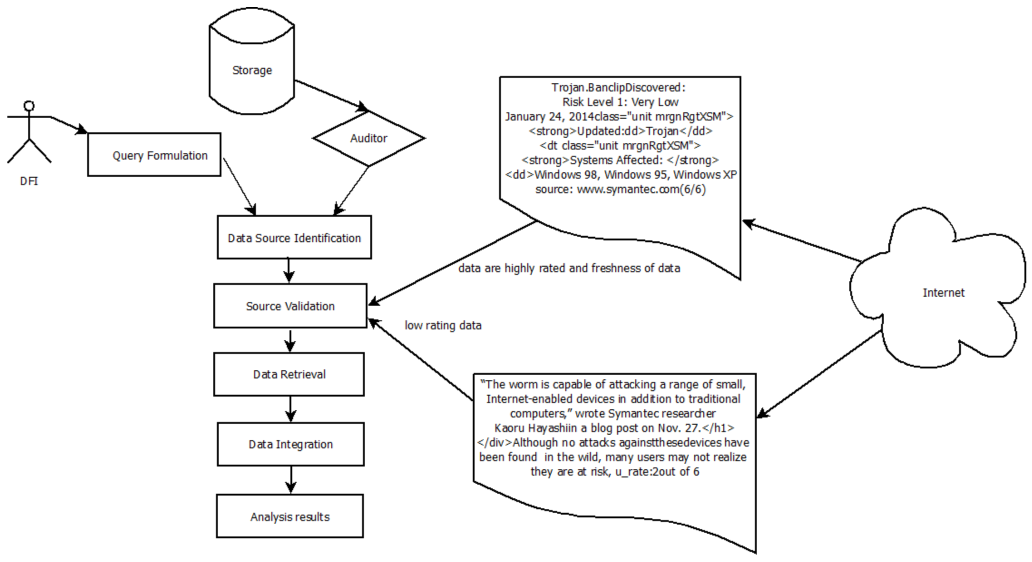

Storage
Auditor
DFI
Query Formulation
Data Source Identification
Source Validation
Data Retrieval
Data Integration
Analysis results
Trojan.BanclipDiscovered:
Risk Level 1: Very Low
January 24, 2014class="unit mrgnRgtXSM">
<strong>Updated:dd>Trojan</dd>
<dt class="unit mrgnRgtXSM">
<strong>Systems Affected: </strong>
<dd>Windows 98, Windows 95, Windows XP
source: www.symantec.com(6/6)
data are highly rated and freshness of data
low rating data
Internet
"The worm is capable of attacking a range of small,
Internet-enabled devices in addition to traditional
computers," wrote Symantec researcher
Kaoru Hayashiin a blog post on Nov. 27.</h1>
</div>Although no attacks againstthesedevices have
been found in the wild, many users may not realize
they are at risk, u_rate:2out of 6

Figure 5.1: Transition of the merge procedure between keystone and cornerstone web data

the sites by individuals who are not that popular are considered to be less important by the algorithm. This process is automatic as the algorithms conduct a watchword based validation of the web pages.

3. ***User rating based:*** Collaborative filtering or recommendation approaches can be used to exploit the unbiased ratings provided by users, that previously acquired that data.

4. ***Freshness of the Information:*** The overall quality of the results is higher if the situational data share some identifiers with the internal cubes or if a corresponding transcoding function is available. If such a transcoding is not available a semi-automatic technique should be adopted.

For cloud security intelligence data discovery, the following sources of cornerstone information are taken under consideration.

1. ***Verified Web Cornerstone:*** The data obtained through the web sites are known as cornerstone. The open cornerstone data are obtained from websites are obtained

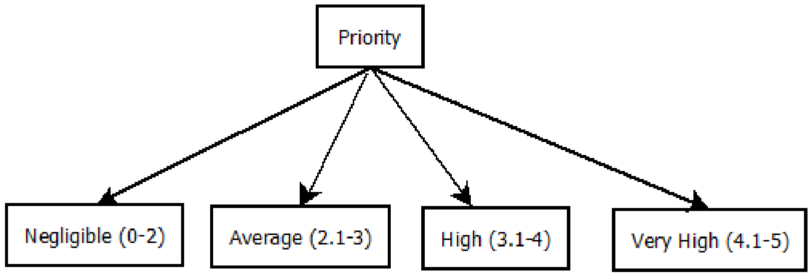


Figure 5.2: Validity Value allocation to the sources of cornerstone data

from anti-virus companies namely Kaspersky and Symantec. The specified websites have been selected mainly due to the data contained in those are similar for answering the research question.

The websites come with high significance factor as the reliability of the information posted on the web pages are verified by the owners. In particular, the data are obtained for free in addition for being accurate. Hence such information play a prominent role in the self service CSI solution proposed here.

2. ***Volatile Blog and Social Cornerstone:*** These sites allow participants to interact and continuously provide data. The data obtained from those provide huge information and reflect the opinions of the people. However the reliability of such data is questionable because the users are not verified. Here the cloud data sources for the specified information are analyzed as a set of input strings.

   Since the integrity of such blog pages are questionable, hence standard scale of 1 to 5 are used for user reviews of the sites. The user ratings are considered to be the underlying benchmarks for CSI in case of the web pages described here.

5.5 Cornerstone algorithm

Algorithm 1 proposed here shows the mechanisms needed to collect cornerstone information from the web. The L_{og} method keeps track of all the web requests conducted by the algorithm in a journal file. The log variable keeps track of all the web requests conducted by the algorithm in a log file. The *Req* variable passes a string that records all the web requests in context to the information required for retrieval. The Obj_{ref} identifies the purpose of target of the cornerstone data,and matches keywords passed to it with the context browsed by the url identifier.

Algorithm 5 Identifying web pages to determine contents of selected URL

```
Output: $Assign = 0$
Input: $MakeReqToUrl$
 1: Begin
 2: procedure CORNERSTONE($L_{og}, Req_m, Obj_{ref}, URL_i$)
 3:     $L_{og}, Req_m \leftarrow SuffTree[leafnode = NULL]$
 4:     $Obj_{ref} \leftarrow *Mem_{L}oc$ and
 5:     $URL_i \leftarrow Locator[stringurl]$
 6:     For (req=1;req¡sum.req;req++)
 7:     while $Name \neq Null$ do
 8:         $self.name = name$
 9:         while $GetUrl \neq URL_{self,name,NULL}$ do
10:             $Error\ stating\ value\ must\ have\ name$
11:             $name = Dictionary[name]$
12:         end while
13:         $HasAttr(Req_m, Message, LevelOfLog = Num_{id})$
14:         $URL_i = URL(UrlIdentifier)$
15:     end while
16: end procedure
```

Initially the range of requests are passed by the digital forensic investigator to the algorithm. Next for all the requests, an object is used to identify the websites denoted as *self.name*. A name is attached to every *self* object. The log method will generate a message with *LevelOfLog* variable used as a number ranging from 1 to 3 together with the crawler's name. The level signifies the number of variations in websites. Currently level

1 identifies whether the data retrieved dealt with a single web sources, 2 signifies multiple sites and 3 signifies social sites together with good general web pages.

5.6 CrawlCornerstone algorithm

Algorithm 2 identifies the procedure such as the *AssignCrawler* takes name of the request and the crawler function as parameter. In addition, it checks whether the url is already in the list of parsed locators as the name would then be presenting the dictionary. If the url is new and not parsed, it is added to the dictionary of names and then passed onto the crawler function for retrieving data.

Algorithm 6 Collection of contents with site-rating

```
1: Begin
2: procedure CRAWLCORNERSTONE($L_{og}, Req_m, Obj_{ref}, URL_i$)
3:     $log(Req_m, Message, LevelOfLog = Num_{id})$
4:     $GenerateMessage(message, crawler = name, level)$
5:     $AssignCrawler(name, crawler)$
6:     while $HasAttr(url) = URL(UrlIdentifier)$ do
7:         $HasName = Dictionary[name]$
8:         return $Cornerstone\ already\ crawled\ to\ \%s\ L_{og}$
9:         $self.name = crawler(name)$
10:    end while
11:    $crawler(self)$
12:    $assertHasAttr(self, crawler, Null)$return $Crawler\ not\ bounded\ \rightarrow\ self.crawler$
13:    while $CurrentDisk \neq NullandCrawlerDisk - CurrentDisk >= 1$ do
14:        $Req_{crawl.self} = Dictionary[url.name]$
15:        $Assign = MakeReqToUrl$
16:        return $Req_{url}$
17:        $URL.filter = FALSE$
18:        $HandleRequest(cls, Req_{crawl}) = ContentsUrl$
19:        return$Contents\ of\ the\ URL\ with\ User\ Ratings$
20:        $CountRequest = CountRequest + +$
21:    end while
22: end procedure
```

The crawler function carries out a check of whether the array to store the retrieved data ave sufficient space on disk. Next it executes a request message that takes the web source name from $Dictionary[name]$. Afterwards it creates an array with cornerstone $id[data]$ and assigns the web page source to the array in string format. Hence the entire available textual data are collected from the web is collected as $assign = MakeReqToUrl$. The stated function returns the request to the server by setting the filter of the url to FALSE.

As identified in the previous chapter, the secured provenance keystone data obtained from the cloud data centers are stored at the Auditor's vm-instance. Considering a digital investigation of the cloud service provider to take place, the DFI will generate a query based on the requirements. The $QueryFormulation$ process identified in Figure 5.1 is the stated function. It is assumed that the query formulation is based on the cloud user requirements and are defined by the user with respect to the standards of the framework of Figure 5.1. After formulation of queries it is passed to the $Auditor$ process, that identifies from it the required keystone data needed for satisfying the query.The auditor process will conduct a watchword based search using Algorithm 3 and return the keystone output.

5.7 AssignValidityValue algorithm

The data stored in unstructured form will be queried irregularly at only specific requirements. Since the data are not queried often and most dimensional methods may have $NULL$ attributes, the system is designed to meet specific query needs. Since the data will be queried independently, the cardinality of the cornerstone will be low. Arranging the data in traditional single table form will predominantly give rose to very high overheads during query processing.

The high overhead has been prevented in the proposed algorithms using causal chaining of data schema, preferably called snowflake schema [6]. The data are divided into facts an dimensions. The facts include attributes of data and the dimensions includes the values or information related to the attributes. To allocate increased provisions for more web data, data dimensions of the cornerstone data are dynamic, harnessing the elasticity of cloud resource provisioning.

Algorithm 7 Allocation of Validation Value based on watchword, user and owner ratings

```
Begin
procedure ASSIGNVALIDITYVALUE($value, key.Rank, user.Rank, cs.Date$)
    $Value = 0$
    $SetOwn[] = Symantec, Kaspersky, AVG$
    while $cs.Date - sys.Date < cur.Date - sys.Date$ do
        $fresh.Date - cs.Date$;
        while $cs.rating >= 4$ do
            $cs.imp = High$
            $user.Rank = 1$
        end while
        while $cs.Keyword = key.Keyword[]$ do
            $key.Rank = 1$
            $own.New = Own[]$
            $o.Rank = 1$
        end while
        $GenerateLevel(dateval, user.rank, key.rank, own.rank, frsh)$
        $seti_{1<=j<5} = dateval + user.rank + key.rank + own.rank$
        while do$GenerateLevel = 3$
            $priority = Moderate$ and $setValue = GenerateLevel$
        end while
        return $GeneralValue$, $o.Rank$ and $user.Rank$
    end while
end procedure
```

5.8 Transition of the proposed model

The data available on the web are present in large volume and the amount of data grows at a rapid velocity. The web data can be used in various situations and are extremely volatile. Those can be referred to as cornerstone data since those can play a significant role in prediction and business decisions. The fact must be addressed that cornerstone data combined with structured and formatted keystone data can provide Security Intelligence

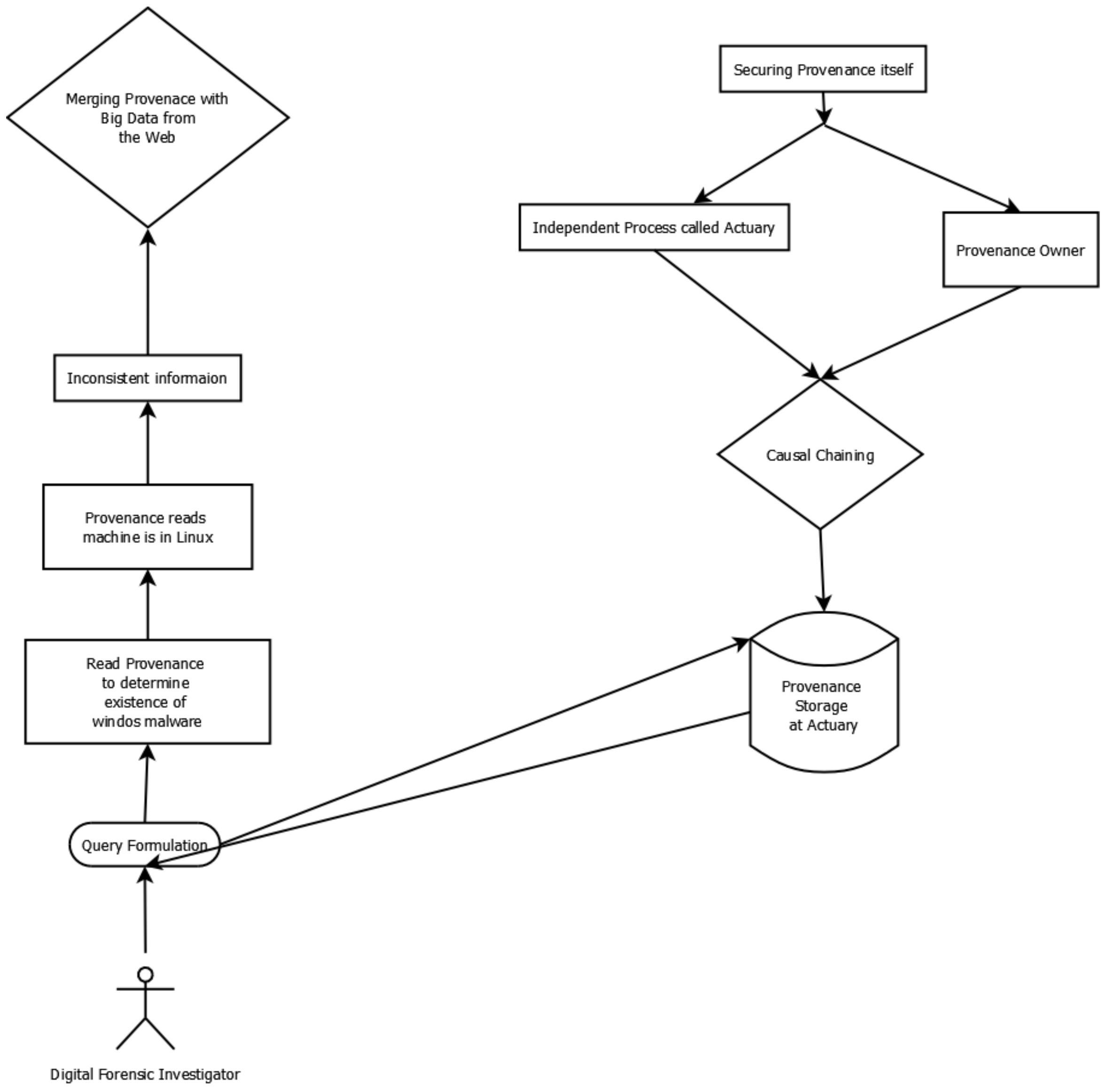
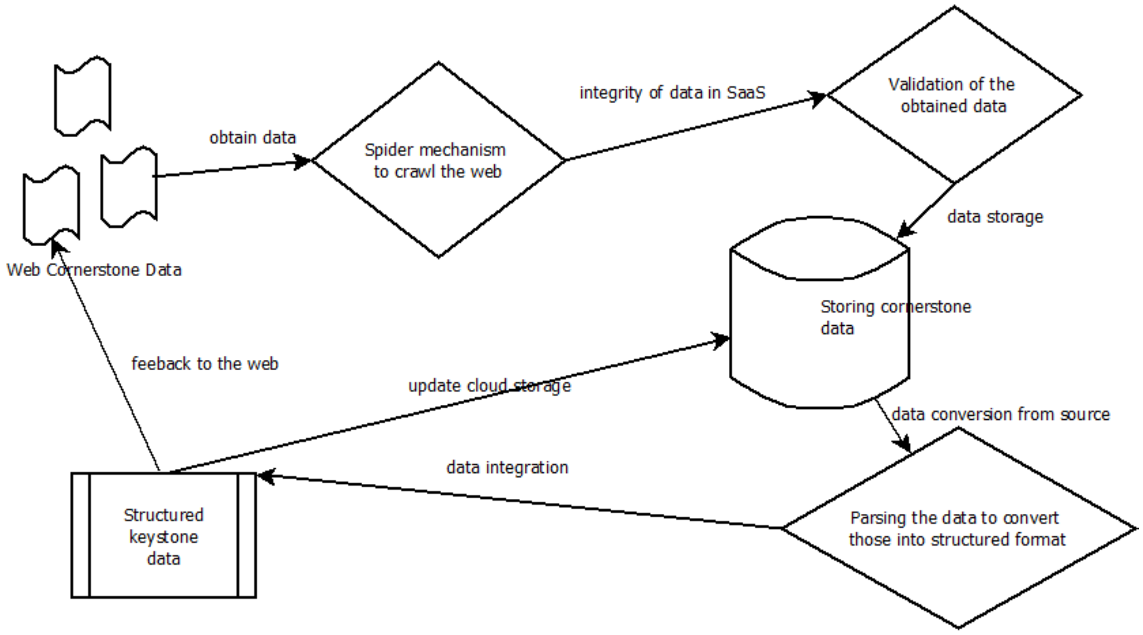

Merging Provenace with Big Data from the Web
Securing Provenance itself
Independent Process called Actuary
Provenance Owner
Inconsistent informaion
Causal Chaining
Provenance reads machine is in Linux
Read Provenance to determine existence of windos malware
Provenance Storage at Actuary
Query Formulation
Digital Forensic Investigator

Figure 5.3: Query formulation to scenario based keystone and cornerstone data integration

(SI) and Security continuity (SC) to a greater degree of accuracy.

Structured data, referred to as keystone data here, can be effectively increased in value if melded with free cornerstone data available on the web. The research challenges lie in obtaining the data from the web, ensuring the integrity and freshness of the obtained data and merging the correct cornerstone data with keystone data for better analytics. The cornerstone data are widely available on the web and specific cornerstone data are used



obtain data
Spider mechanism to crawl the web
integrity of data in SaaS
Validation of the obtained data
data storage
Web Cornerstone Data
Storing cornerstone data
feeback to the web
update cloud storage
data conversion from source
data integration
Structured keystone data
Parsing the data to convert those into structured format

Figure 5.4: Transition of amalgamation between keystone and cornerstone data

to mitigate special user needs. Based on the requirements of accuracy, the importance of melding keystone and cornerstone data is manifold. There is an increasing research need to amalgamate cloud keystone provenance and cornerstone data to achieve high accuracy SI in a short length of time.

The real life use of cloud infrastructure called OpenStack has been used to manage the cornerstone data. Next effective algorithms for mining the data can be conducted in the cloud environment harnessing its dynamic nature and capability to contain large volume of data with great variety.

The research goal is to propose novel algorithms for parsing cornerstone and ensuring the accuracy of those. Secondly effective algorithms to conduct keyword based structuring of the cornerstone data will be proposed. Thirdly algorithm for determination of the reliability and integrity of the cornerstone data will be proposed based on owner ratings, user rating and freshness of information. Finally, results of merging cornerstone data with keystone provenance are analyzed and performance compared for those.

As stated, using web data in cloud after detection have been addressed as the research issue. The results of the earlier experiments gives rise to another issue of analyzing provenance data through fusion with upto date unstructured data obtained through the World Wide Web for accurate Security intelligence (BI). The integrity of the web data must be ensured, and due to its unstructured format and large volume, it is often referred to as Web Cornerstone Data. Here the structured provenance data obtained from the system are referred to as keystone data and the unstructured data obtained in large volumes from the web are referred to as cornerstone data.

Even if the keystone data stored in data warehouses are analyzed by the mechanisms stated in the previous research, rapid decisions have to be made in times of business threats based on reliable data, and evaluations of potential consequences have to be made harnessing the power of the cloud. Hence, the necessity of fusing keystone with cornerstone data will enrich the knowledge base of the Digital Business Prediction manifold and increase the certainty of achieving accurate BI.

Previously, analysis of data in multi-dimensional Data Warehouses have been discussed in [1]. Those data can be queried using Online Analytical Processing (OLAP) strategies to reduce overhead in terms of query execution time. Here the use of OLAP to query Keystone data is is ensured and the performance compared with traditional query processing. OLAP will enable interactive queries to be executed and upon merging those with web data, effective decisions can be taken by the Digital Business Predictions.

Static multi-dimensional provenance is identified as keystone data due to its tamperproofness and reliability achieved through the solution of the previous research [2], [3]. This research proposal targets to fuse keystone data with volatile unstructured data called cornerstone. Cornerstone data are obtained from the web, and the name is given mainly due to the criticality, unstructured format and volatility of those. Outcome of this research will contribute to the field of SI and ensure proper SC through cloud and cornerstone data.

5.9 Mathematical model of cloud provenance detection and web cornerstone data amalgamation

This section identifies the mathematical model of real-life scenarios that are used to conduct the experiment in this research. The system S$_n$ consists a finite set of real life application A$_i$ where A$_i$ = $\left(a_1,\ \ a_2,\ \ \cdots\ \ ,a_u\right)$. Each application a_i consists of a finite set of processes P$_j$ denoting set of all processes such that the condition $\forall a \in A : P_j = \left(p_1,\ \ p_2,\ \ \cdots\ \ ,p_v\right)$ is satisfied where u and v are positive integers. Each process p$_c$ have states that are represented by a finite set $S_t = \left(s_1,\ \ s_2,\ \ \cdots\ \ ,s_n\right)$. The states are divided into two subsets representing the correct and cloud keystone sets and cornerstone sets respectively, where the value is obtained by S$_{correct}$ = $\left(s_{l1},\ \ s_{l2},\ \ \cdots\ \ ,s_{ln}\right)$ and incorrect state as S$_{CloudKeystone}$ = $\left(s\prime_{l1},\ \ s\prime_{l2},\ \ \cdots\ \ ,s\prime_{ln}\right)$. Hence the above cases are true if and only if

$$(S_{cornerstone} \bigcup S_{keystone} \in S_t) \wedge (S_{cornerstone} \bigcap S_{keystone} \in \emptyset)$$

are true. The combined states of the various processes in the real life applications running on the cloud platform is satisfied by the equation 5.1.

$$\begin{aligned} B_i = (\forall i \in 0 < i \leq n : b_i \subset S_{cornerstone}) \wedge \\ (\forall j \in 0 < j \leq n : b_j \subset S_{keystone}) \end{aligned} \tag{5.1}$$

Valid write counts is projected as $(s_l, s\prime_l) : S_t \in \delta_p \wedge s_l, s\prime_l \in S_t$. Here δ_p is the probability of the validity of cornerstone data state. Cornerstone data have a validity value of 5 if and only if a process p$_j$ executes operation w$_j$ on the file that is from a verified website like Symantec. Hence, if $w_j\prime$ is a set of cornerstone data operations that result in validity value to be less than 3, we get,

$$w_{j}\prime = (s_l, s_l\prime) : (\exists p_j : x \notin w[j] \wedge x = S_t) \tag{5.2}$$

Keystone is already considered to be in valid state since the data are operations can be carried out in multiple files. Hence the files that are in specific group denoted by G(R$_k$) is described here.

$$\begin{aligned} G(R_k)(s_l, s_l\prime) = ((s_l\prime) : \forall y : y \in r_y : v(s_l) \\ \neq v(s_l\prime) \vee v(s_l\prime) = v(s_l) \vee \forall y : y \notin r_y : v(s_l) \\ = v(s_l\prime) \wedge v(s_l\prime) \neq v(s_l)) \end{aligned} \tag{5.3}$$

$$\begin{aligned} \exists p \in P_j : G(s_l) \subseteq S_t \vee G(s_l) \subseteq S\prime_t) \\ \implies G(s_l) \subseteq R_x \wedge G(s_l) \subseteq R\prime_x = \emptyset \vee \\ G(s_l) \subseteq W_y \wedge G(s_l) \subseteq W\prime_y = \emptyset; \end{aligned} \tag{5.4}$$

An application a$_i$ $(0 < i \leq u)$ consists of a process p$_j$ $(0 < j \leq v)$ that carries out website cornerstone read and write operations on the data stored in the cloud. To ensure low latency fault detection by proposed architecture, faults must be detected with minimal delay in which those occur [2]. Website read operations are represented by the set R$_x$ and write operations are represented by the set W$_y$. After read or write operation by p$_j$, the state is saved in a state variable s$_l$ and updated in S$_t$. If the state has low validity value the S$_t$: s$_l$ = $s\prime_l$ else S$_t$: s$_l$. Hence, the conditions need to satisfied if s$_l$ is to have high validity value and hence the cornerstone data are reliable.

5.10 Summary

This chapter aims to merge provenance keystone data with unstructured cornerstone data from the web to improve the accuracy of cloud security intelligence. The importance of merging keystone and cornerstone data have been highlighted. At the same time, the attributes of keystone and cornerstone have been identified to differentiate between the two types of data. Next the methodology of the proposed scheme and the validation metrics and criteria have been proposed here.

Algorithms 1, 2 and 3 in this chapter identify the crawling of web pages, obtaining data from selected web pages and validating those for use with keystone data. Next the algorithms for converting unstructured data into structured format and forming a fusion cube with keystone data for security intelligence have been proposed. The mathematical analysis of provided in this chapter evaluates the ability of the proposed mechanism to improve the accuracy of secure provenance based integrity assurance in cloud computing environments.

Chapter 6
Result Analysis

This chapter identifies the effect of the proposed provenance securing and analysis schemes through experimentation on real life cloud service environments. Performance benchmarks in terms of CPU overhead, delay, vm-counts and file sizes have been analyzed. Global Delay (TGD), Inter-message Delay (IMT) and Number of Retries for securing cloud provenance have been analyzed upon comparison with existing benchmarks identified in [48]. Real life scenarios for the collection for provenance information namely file read, write and copy operations have been tested during experimentation and results obtained for those.

Tests were conducted for the collection of provenance in a secured procedure and using a third-party *Auditor* process for the security scheme. This involved the proposed *ProvOCal* algorithm for capturing the cloud provenance and the *ProvCapsule* algorithm for binding the provenance with the system data. Additionally the memory and disk locations of the physical machines in the cloud storing specific provenance information were also identified for increased security. Securing the provenance using the proposed mechanism resulted in system delay and decreased throughput. Additionally frequencies and cumulative frequencies of the total number of retries for specific ranges of delay caused by securing the provenance have been analyzed as well using standard formulae identified in [21].

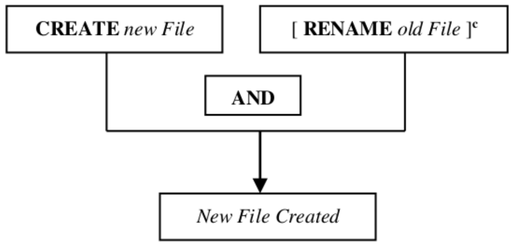
After analysis of the provenance securing scheme, experimental results have been represented graphically for the algorithms proposed for web cornerstone data amalgamation with provenance to achieve increased *SI*. The cornerstone data are crawled, converted to structured format and merged with provenance information as discussed in the previous chapters. Next, data were collected for six real life applications running on pre-specified number of vm's identified by vm-ids in OpenStack Essex cloud on Ubuntu 12.04 LTS servers. The accuracy of the amalgamation and the increased *SI* achieved through merging cornerstone and keystone provenance information have been presented. The rest of the chapter discusses in detail about the obtained results.

6.1 Sample Scenarios subject to Secured Provenance Detection

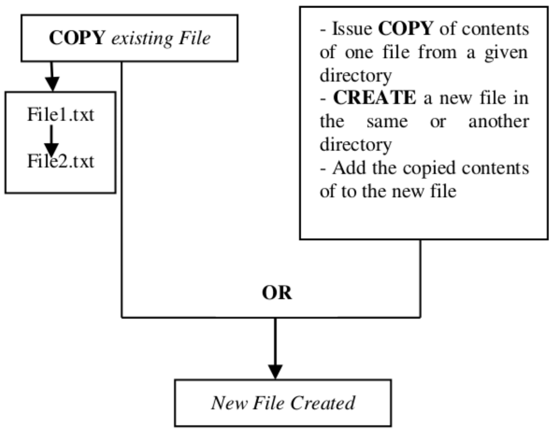
The experiments were conducted in real life cloud environment running OpenStack cloud in Ubuntu 12.04 (Long Term Service) LTS servers. Total of 10 nodes were involved each with 32 GB of Random Access Memory (RAM), 2 Terabytes (TB) of hard drive and core i7 processors. Kernel Virtual Machine (KVM) was used for the cloud, that enabled obtainment of 20 vCPUs from each CPU and memory was shredded in different flavours in accordance to the requirement of the cloud vm-instances. Next the scenarios for which the provenance was collected for the system level was identified. In an Operating System each event is divided into a series of atomic steps. Each of the steps can be derived from the atomic actions and the kernel system calls. A specific pattern must be followed [21], [48]. This pattern is called the signature of the operation which is necessary for provenance detection since it characterizes and provides behavioral information of the activity of that process.

Analyzing and identifying signatures of text based file operations is a prime goal of provenance detection of file activities in cloud computing environments. Signatures of text oriented file system calls of Linux Operating Systems are listed below.

- Issue command to **CREATE** text file in a pre-specified process and directory **AND**
- The **RENAME** operation is **NOT** executed on the same File in the same process, i.e, [**RENAME** old File]



(a) Provenance Capture of File Creation and Delete



COPY existing File
File1.txt
File2.txt
- Issue COPY of contents of one file from a given directory
- CREATE a new file in the same or another directory
- Add the copied contents of to the new file
OR
New File Created

(b) File Copy Operation at the Activity Layer

Figure 6.1: Provenance cognition from system centric operations for file creation and write at the activity layer

- Issue **COPY** command to the system call, **OR**
- **CREATE** a new File in the same or new directory and **COPY** the contents of the old File and paste them on the new File.

At the Data Layer, the main objective is to analyze the logs which are collected at the System levels [22], [26]. Log files of Cloud Computing provide information that can be used to collect end-to-end system provenance. The provenance information collected in this way will be highly useful to achieve security and trust on behalf of the cloud customers from the cloud service providers.

Based on the analysis of the cloud computing infrastructure and existing provenance models, provenance detection algorithms must be placed forward with an aim of detecting file operations which have possibility of enabling data leakage. *ProvCapsule* and *ProvOCal* are such algorithms that aim to drive real time provenance from cloud activity layer without hampering the fault tolerance of the cloud.

The overhead incurred for encapsulating provenance in terms of global delay and message transmission delay needs to be measured for the algorithms proposed here. The results

Table 6.1: Global Delay, Inter-Message Delay (IMT) and Retries of provenance capture

VM-Count	Size	TGD(s)	IMT(s)	Retries
140	512	1817	28.9	0
100	1024	1603	91.4	0
160	1536	1533	90.9	0
162	2048	1470	102.7	7
184	2560	1389	169.3	3
190	3072	1118	281.61	7

Table 6.2: Overhead of the proposed model as compared to earlier mechanisms

Model	VM-C	Size	OvhdRAND(%)	OvhdSEQ(%)	OvhdACC(%)
ProvCapsule	200	512-1024	2.44	1.13	0.80
	200	2048-2560	0.66	0.48	0.24
SecLaaS[49]	200	512-1024	4.12	2.06	NA
	200	2048-2560	1.88	1.37	NA

of the algorithms must be compared to established benchmarks to ensure that the overhead incurred does not exceed the pre-defined limit [48]. The following sections identify the performance of the algorithms in terms of overhead incurred for $Transmission\ Delay$ (TD) and *Global Delay* (*GD*) for encapsulating provenance.

6.2 Analysis of results for the securing mechanism

The results of overhead calculation for the proposed algorithms are tabulated in Table 6.1 and Table 6.3. The overheads of the proposed *ProvCapsule* with *SecLaaS* [49] has been shown in Table6.2. The number of vm-counts that transferred files of specific sizes ranging from 512-3072 MB are shown in column $VM - Count$. The Total Global Delay TGD is shown for the specific vm-instance counts in seconds. In addition the $Inter - Message$ $Transmission\ Delay$ ($IMTD$) IMT are also shown in seconds together with the number of retries of provenance encapsulation.

Table 6.3: Mean Delay and Standard Deviation of Provenance Encapsulation

Size	M.Cont.	M.Time(s)	M.Delay(s)	STD.Dev
512	5.104	6.042		
100	5.924	8.584		
160	6.368	7.505	8.198	1.434
162	7.011	8.866		
184	8.026	9.990		

Table 6.4: Frequencies and Cumulative Frequencies of different Delay Ranges

Delay Range	Frequency	Cumulative Freq.(%)
1 - 2	14	1.50
3 - 4	22	2.40
5 - 6	104	11.1
7 - 8	422	45.1
9 - 9.9	374	40.0

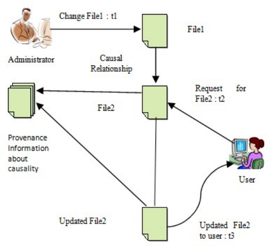
The algorithms were implemented in the cloud controller that hosted 936 vm-instances. Hence there were a finite population of vm-instances X_i such that $X_i = \left(x_1, \quad x_2, \quad \cdots \quad ,x_n \right)$. Since there are a finite population of vm-instances so the $Mean\ Delay$ incurred by all those is finite as well. Hence we detect Mean, Variance and Standard Deviation for any X_i as,

$$\overline{X_i} = \frac{\sum_{i=1}^{n} X_i}{n} \tag{6.1}$$

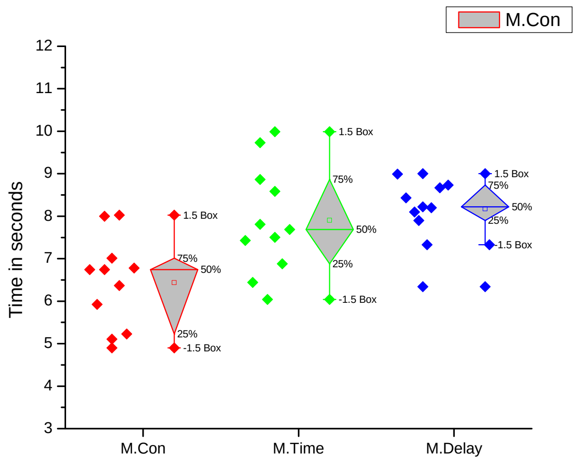
The value can be obtained for a large number of observations as denoted by n. The variance of the delays in different ranges of Global Delays (TBD) and Inter-Message Time (IMT) is given by,

$$S^2(n) = \frac{\sum_{i=1}^{n}(X_i - \overline{X(n)})^2}{n-1} \tag{6.2}$$

The closeness of the variance S_n^2 to mean μ can be determined as $Var(\overline{X(n)})$ is the ratio of total variance and number of occurrences for a given period t_i.



(a) Management of Provenance Information in the Cloud



(b) Global Delays and Inter-message Delay Distributions

Figure 6.2: Provenance cognition from system centric operations

$$STD.dev = \sqrt{\frac{\sum_{i=1}^{n}(X_i - \overline{X(n)})^2}{n(n-1)}} \tag{6.3}$$

Table 6.3 highlights the Mean Time ($M.Time$) required for provenance encapsulation for different sized files and different counts of vm-instances. The $M.Time$ is shown to be less than 10 seconds for 936 vm-instances. As identified in [23], the benchmark of tolerable time for provenance encapsulation is 10 seconds for a large system. Taking that value as a benchmark, implementation of *ProvCapsule* and *ProvOCal* algorithms show that the time overhead is below 10 seconds for over 900 instances. The overhead is found to be 8.198 seconds on average which is acceptable. The standard deviation is found to be 1.434 which is desirable.

An important aspect of provenance management regarding data leakage is the causality of information. Information from one process can be causally related to information of another process, hence the atomic actions can be causally combined to evaluate against pre-specified benchmarks and reduce false-positives.

Finally Table 6.4 shows the delay range and frequency of delay for all the vm-instances. It is seen that 45% of the instances face an average delay of 7-8 seconds. The delay between 9-9.9 seconds is faced by 40% of the instances for encapsulating provenance metafile. Hence the average delay is below 10 seconds for both the algorithms.

The variance and the distribution of the vm-instance classes are shown in Figure 6.3. The variance is maximum for $M.Con$ whereas it is minimum for $M.Delay$ since the resources allocated in terms of memory and storage for each instance class was different. Hence the benchmark stated in [23] is satisfied.

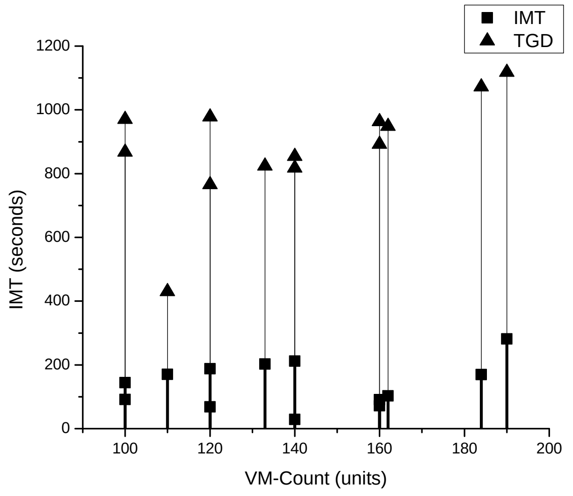
6.3 Analysis of Results for Merging Cornerstone and Keystone Provenance

The performance of the proposed algorithms were tested and analyzed in real life cloud environments running OpenStack Essex cloud. Six commercial real life applications provided as Software as a Service (SaaS) in cloud vm-instances were analyzed at the servers of the cloud service provider. The SaaS applications included Inventory Management Software (IM), Accounting Management Software (AM), Human Resource Management Software (HRM), Office Document Management Software (ODM), Sales Management Software (SM), Customer Relationship Management Software (CRM) as identified in Table 6.5.

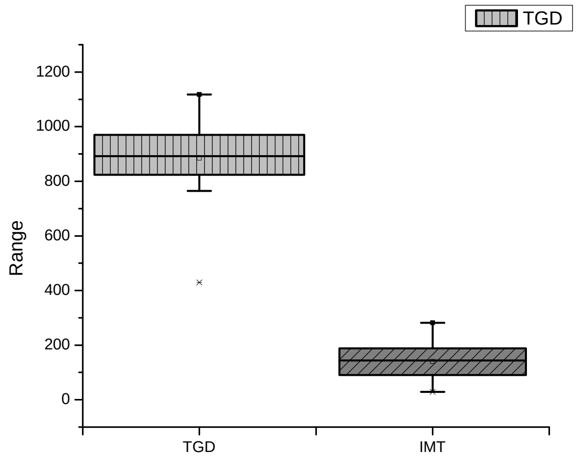
There are a finite population of vm-instances so the number of accepted requests is finite as well. Hence we detect mean acceptance rate for any X_i as,

$Acc_i = \left(x_1, \quad x_2, \quad \cdots \quad , x_n \right).$

$$\overline{Acc_i} = \frac{\sum_{i=1}^{Count} Acc_i}{Count} \tag{6.4}$$



(a) TGD and IMD counts



(b) Distribution of overhead

Figure 6.3: Results of provenance cognition using the proposed algorithms

Table 6.5: Performance evaluation of application level provenance cognition

App.	VM-id	Case	Alg Det.	SR (%)	MR (%)
IM	10	1084	922	85.0554	14.9446
AM	13	1001	968	96.7032	3.2968
HRM	07	834	772	98.3871	1.6129
ODM	16	896	842	93.9732	6.0268
SM	22	1200	966	80.5000	19.5000
CRM	20	942	800	84.9257	15.0743

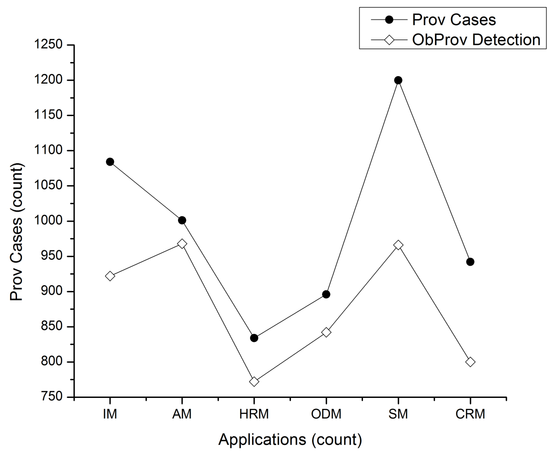


Figure 6.4: Sequence of transitions and checks from one state to another in the proposed architecture

Table 6.6: Mean Rejects and Cumulative frequency values for *ObProv*

Req. Size	Count	Acc.	Rej.	M.Delay	C.freq
10-100	63	13	50		
101-200	74	4	70		
201-300	69	20	49	64	21.43
301-400	82	18	64		
401-500	69	10	59		

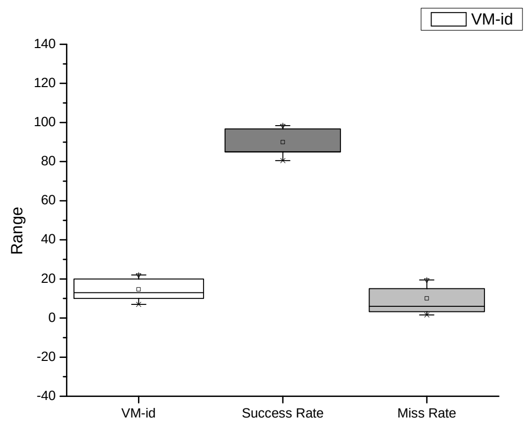


Figure 6.5: Sequence of transitions and checks from one state to another in the proposed architecture

The value can be obtained for a large number of observations n. The variance of the delays in different ranges of $Total\ Global\ Delays\ (TBD)$ and $Inter-Message\ Time\ (IMT)$ is given by,

$$S^2(n) = \frac{\sum_{i=1}^{Count}(Acc_i - \overline{Rej(Count)})^2}{Count - 1} \tag{6.5}$$

Individual vm-instances were allocated for each of the SaaS applications specified and provided to cloud customers. Each customer has multiple users accessing the applications with their specific user names and passwords. For each customer, specific provenance information were collected that includes id-used to access, time and date of access, pages visited and changes made to files in terms of write operations. The provenance were detected from the SaaS level using *ObProv* algorithm and and stored in a separate cloud server running the $Auditor$ algorithm for a period of 6 months to ensure that the data are in large volume, increased value and velocity, thereby satisfying the three v's of web cornerstone data.

The captured provenance using *ObProv* algorithm were stored in Cloud Provenance server that as $Auditor$ algorithm implemented on the machine using Linux platform and PostgreSQL database was used as it was already configured in the data center of the cloud service provider where the experiments were conducted. The $Auditor$ algorithm provides authorized access to the provenance information based on timestamp authentication, thereby enabling the prevention of undesired access that would otherwise obtain the provenance files and tamper those [50]. Hence $Auditor$ algorithm ensures tamper-proofness of provenance information. Table 6.5 highlights the results of provenance access acceptance and rejections by $Auditor$ algorithm.

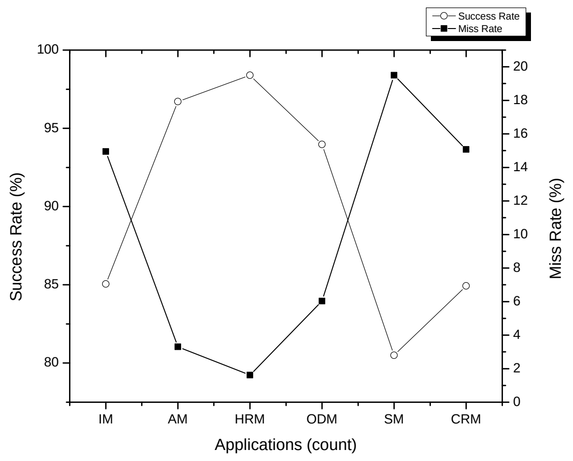


Figure 6.6: Sequence of transitions and checks from one state to another in the proposed architecture

6.4 Discussion of Results

Collection of provenance from the application layer are obtained for six real life applications provided as Software as a Service (SaaS) already identified in the previous chapter. The applications are used in real life by different number of users and provenance were collected from those in terms of operation executions, file accesses times and file user's identity. Next the system and application level provenance keystone data were secured by the *PO* and passed to the independent *Auditor* process.

The *Auditor* process further encrypted the provenance with its private key, as a result the causal chain of provenance security was used instead of onion framework as it would prevent the top level of encryption from being exposed because the second level of encryption is conducted by a different body than the first layer. At the same time the processes executed in the cloud are dynamic and those occur in real-time. Hence the causal chain mechanism will have less overhead as found in the research because of its less layer of encryption. The onion framework provides increased security since it implements atleast 4 layers of encryption, however the overhead incurred in terms of CPU cycles and turnaround time is significantly higher for the onion framework as compared to the causal chain. Hence the causal chain framework is more applicable for the cloud due to its low CPU overhead and adaptability to rapid response that is desirable for the cloud system.

Web cornerstone data amalgamation results highlight the success at which unstructured cornerstone web data can be structured using watchword matching and fitted with keystone provenance. Six real life applications running as SaaS were monitored and provenance information were collected from those. Security cornerstone data were collected from malware and anti-virus websites as stated previously in the research.

Confidence on the cornerstone data were determined on the basis of user rating as specified in [16]. Next the unstructured data were converted to structured format on the basis of watchword matching and stored in database using multi-dimensional *Spider Schemas*, giving rise to a fusion-cube. Chapter 4 identified the validity values from 1 to 4 assigned to the data obtained from specific web sources based on freshness, applicability in cloud security intelligence, user ratings and owner ratings. The Success Rates (SR) for structuring and melding the cornerstone data to security keystone provenance of the six specific SaaS applications are found to be 85.0554%, 96.7032%, 98.3871%, 93.9732%, 80.5000% and 84.9257% respectively.

6.5 Summary

The experimental environment for securing provenance into keystone data proposed in this thesis have been depicted in this chapter. At the same time the test-bed for the amalgamation of provenance keystone with cornerstone data have been described. Effective Descriptive Set theory was implemented and Active-Threading was used for the proposed securing scheme. Real life scenarios were depicted and the experiments were conducted a the servers of commercial cloud service provider. Next, Spider Schema was proposed for amalgamation of cloud provenance keystone with web cornerstone data. The result of the proposed model yielded a fusion cube that consisted of multi-dimensional Spider Schema.The fusion cube comprised of cornerstone data and keystone information.

Graphical result analysis showed that the mean delay incurred for the proposed provenance securing schema is 8.1% that is desirable. For provenance amalgamation and fusion cube formation, the Mean Success Rate (*MSR*) was found to be 89.33%. At the same time the *MR* was below 9.67% that is below the benchmark of 10.00%, yielding suitable results. The reason for the desirable *SR* is due to the web page rating mechanism on the bass of data freshness, user ratings and owner ratings as proposed in this thesis. Hence the proposed models and the algorithms of those have been validated and established in real life production level cloud environment.

Chapter 7

Conclusion

This chapter highlights the research findings and emphasizes the effectiveness of the proposed methodologies to secure cloud provenance and amalgamate web cornerstone data with those for improved Digital Forensic Investigation (*DFI*). The two contributions of securing the keystone provenance to ensure tamper-proof storage of those and integration of keystone and cornerstone data with an aim of giving system wide view to digital forensic investigations to enable correct auditing, are reviewed here. Results show that the performance of the proposed algorithms improves the existing research contributions in terms of overhead for random and sequential writes. At the same time, the overhead of real life applications for accessing provenance journal files on the servers of cloud service providers are presented as additional performance analysis.

The future work of the research has been highlighted to identify the scope of improvement of this thesis. Firstly, the scope of future research in cloud provenance has been highlighted. Secondly, the scope of future work in the cloud provenance keystone security and cornerstone amalgamation has been discussed in the following sections.

7.1 Securing Cloud Provenance using Causal Chaining an Independent Auditor process

The detected cloud provenance have been dynamically secured using private key of the Provenance Owner (PO). Next, Active-Threading mechanism was used to transmit the provenance and establish causal relationship among those through time-stamp aggregation. Afterwards, the provenance was transmitted to the independent Auditor process that is verified by both the PO and the Cloud Customer.

The Auditor carries out a single level of encryption with its private key. Finally, both the PO and the Auditor algorithms share the public keys of those. During Digital forensic Investigation (DFI), the provenance stored at the Auditor is decrypted using the public keys and then used by the forensic investigator for analysis.

The proposed models were implemented in OpenStack cloud environment of real life cloud service provider. Ubuntu 12.04 Long Term Service (LTS) was used for the cloud compute nodes and Kernel Virtual Machine (KVM) was implemented to launch 400 virtual machine (vm) instances for conducting the experiments. Results of the proposed algorithms present desirable Global Delay of 8.198 seconds for transmission of file sizes ranging from 512-2560 MegaBytes. Comparison of the obtained results with SecLaaS [49] shows.. that the proposed model outperforms SecLaaS for both random and sequential writes. The overhead of the sequential write of filesizes ranging from 512 to 1024 MegaBytes are found to be 1.13% and 2.06% for ProvCapsule and SecLaaS respectively, highlighting the desirable performance of the proposed *ProvCapsule*. For increased file size of 2048 to 2560 MegaBytes, ProbCapsule has an overhead gain of 1.16% over SecLaaS [49] for random writes.

In addition to random and sequential writes, this research also estimated the overhead associated with access to the provenance journals in the cloud. The access to those in a dynamic system like the cloud is important due to the involvement of a large number of physical and virtual machines. However SecLaaS [49] did not estimate the overheads incurred due to increased access time required to retrieve and secure the provenance. Desirable overheads of 0.80% and 0.24% were obtained for small and large file sizes respectively.

7.2 Amalgamation of Provenance with Unstructured Web Data

Effective DFI using only provenance stored in the data warehouses of the Auditor is not possible, since it requires additional and upto date information for its decisions. Hence, besides tamper-proofing provenance, it must be amalgamated with web data for correct DFI decisions during auditing. Integration of structured provenance with unstructured data in the cloud presented some new challenges such as data obtainment, ensuring validity of web data, structuring web data based on taxonomy and amalgamation of those to structured provenance. The secured provenance was stored in multi-dimensional Spider schema that is capable of declaring new dimensions based on properties of obtained data through watchword matching.

In this research, a model has been proposed to meld keystone provenance with cornerstone data. Algorithm called *CrawlCornerStone* was used to obtain data from the web at high velocity. The *AssignValidityValue* was used to determine the ratings of the web pages based on user rating, owner rating and freshness. Next, the taxonomy of the web data was used to merge with the provenance stored in a multi-dimensional spider schema to form a fusion-cube. This enabled the merging of additional cornerstone with provenance stored in spider schema. Watchword-oriented approach was used for the Taxonomy engine that implemented full-text index and keywords with an attempt to classify different web data documents into coherent taxonomies. Individual IDs were assigned to different taxonomies that enabled correct identification of those for integration with provenance.

The algorithms were tested on six real life Software as a Service (SaaS) applications for experimentation and the Success Rate of cornerstone and keystone integration were calculated. Results show desirable Success Rate (SR) of 89.33% and additionally the Miss Rate (MR) was determined to be 8.66%. The achieved results reflect the effectiveness of watchword and taxonomy oriented cornerstone integration with keystone.

Algorithms namely *ProvCapsule*, *ProvOCal*, *ObProov* and *Auditor* have been proposed for capturing, encapsulating and securing the cloud computing provenance that is also referred as keystone data. *CornerStone*, *CrawlCornerStone* and *AssignValidityValue* algorithms are responsible for obtaining cornerstone data from web, convert those to structured format, identify validity values of the obtained data and form a fusion cube using spider schema respectively. Results have been determined in terms of delay for the securing algorithms together with CPU overhead. The evaluation of the data amalgamation algorithms have been done using real life SaaS applications and scenarios as benchmarks. Desirable results have been obtained for both the research issues based on industry benchmarks.

7.3 Future Work of Cloud Provenance

As stated earlier, the proposed algorithms detects, bounds and secures provenance in the cloud to ensure that those are tamper-proof. Additionally web cornerstone data from the web are used together with system and application level provenance for increased *CSI*. However, performance evaluation using tools such as *OpenR* for data amalgamation analysis is an area of future research.

The research proposed solutions for the identified research questions through implementing watchword matching, effective descriptive set theory, active-threaded provenance security and fusion cube for securing cloud provenance and amalgamate cornerstone data with those. With respect to the above limitations, scope of future research lies in performance evaluation of the proposed schemes using the mentioned tools.

Programs analysis for the algorithms should be carried out to obtain the time and memory complexities and aiming to improve the duration and storage multiplicities of those through multi-dimensional database such as SparQl. Using Online Analytic Processing (OLAP) mechanism for analysis of web cornerstone data and establishment of a new open source cloud computing paradigm by the name of Open Cloud Integrity as a Service (OCI-aaS) is an area of future research interest. Development of mechanisms to increase the learning capacity of the algorithms for web cornerstone amalgamation with keystone provenance and propose the novel Cloud Forensic Investigation as a Service (CFIaaS) is a topic of future research engagement.

Bibliography

[1] Asif Imran, Alim Ul Gias, Rayhanur Rahman, and Kazi Sakib. Provintsec: a provenance cognition blueprint ensuring integrity and security for real life open source cloud. *International Journal of Information Privacy, Security and Integrity*, 1(4):360–380, 2013.

[2] Asif Imran, Alim Ul Gias, Rayhanur Rahman, Amit Seal, Tajkia Rahman, Farhan Ishraque, and Kazi Sakib. Cloud-niagara: A high availability and low overhead fault tolerance middleware for the cloud. In *Computer and Information Technology, 2013 International Conference on*, pages 164–170. IEEE, 2013.

[3] Asif Imran, Emon Kuman Dey, Kazi Sakib, and Abdullah-Al Wadud. Active-threaded algorithms for provenance cognition in the cloud preserving low overhead and fault tolerance. In *Computer Engineering and Applications (CEA), 2014 International Conference on*, pages 249–255. WSEAS, 2014.

[4] Srinivas Krishnan, Kevin Z Snow, and Fabian Monrose. Trail of bytes: New techniques for supporting data provenance and limiting privacy breaches. *Information Forensics and Security, IEEE Transactions on*, 7(6):1876–1889, 2012.

[5] Himel Dev, Tanmoy Sen, Madhusudan Basak, and Mohammed Eunus Ali. An approach to protect the privacy of cloud data from data mining based attacks. In *High Performance Computing, Networking, Storage and Analysis (SCC), 2012 SC Companion:*, pages 1106–1115. IEEE, 2012.

[6] Josiah Dykstra and Alan T Sherman. Acquiring forensic evidence from infrastructure-as-a-service cloud computing: Exploring and evaluating tools, trust, and techniques. *Digital Investigation, Elsevier*, 9:90–98, 2012.

[7] Manish Kumar Anand, Shawn Bowers, and Bertram Ludascher. Provenance browser: Displaying and querying scientific workflow provenance graphs. In *Data Engineering (ICDE), 2010 IEEE 26th International Conference on*, pages 1201–1204. IEEE, 2010.

[8] Kai Hwang and Deyi Li. Trusted cloud computing with secure resources and data coloring. *Internet Computing, IEEE*, 14(5):14–22, 2010.

[9] Alberto Abelló, Jérôme Darmont, Lorena Etcheverry, Matteo Golfarelli, Jose-Norberto Mazón, Felix Naumann, Torben Pedersen, Stefano Bach Rizzi, Juan Trujillo, Panos Vassiliadis, et al. Fusion cubes: Towards self-service business intelligence. *International Journal of Data Warehousing and Mining (IJDWM)*, 9(2):66–88, 2013.

[10] W.C. Tan. Provenance in databases: Past, current, and future. *IEEE Data Engineering Bulletin*, 30(4):3–12, 2007.

[11] R.K.L. Ko, P. Jagadpramana, M. Mowbray, S. Pearson, M. Kirchberg, Q. Liang, and B.S. Lee. Trustcloud: A framework for accountability and trust in cloud computing. In *Services (SERVICES), 2011 IEEE World Congress on*, pages 584–588. IEEE, 2011.

[12] M. Mowbray, S. Pearson, and Y. Shen. Enhancing privacy in cloud computing via policy-based obfuscation. *The Journal of Supercomputing*, 61(2):267–291, 2012.

[13] C. Wang, S. Chen, and J. Zic. A contract-based accountability service model. In *Web Services, 2009. ICWS 2009. IEEE International Conference on*, pages 639–646. IEEE, 2009.

[14] Imad M Abbadi and John Lyle. Challenges for provenance in cloud computing. In *3rd USENIX Workshop on the Theory and Practice of Provenance (TaPP11). USENIX Association*, 2011.

[15] A. Haeberlen. A case for the accountable cloud. *ACM SIGOPS Operating Systems Review*, 44(2):52–57, 2010.

[16] Paul Groth and Luc Moreau. Recording process documentation for provenance. *Parallel and Distributed Systems, IEEE Transactions on*, 20(9):1246–1259, 2009.

[17] R Sunitha et al. Data provenance verification for secure hosts. *International Journal of Information Technology & Computer Sciences Perspectives*, 2(1):180–182, 2013.

[18] Josiah Dykstra and Alan T Sherman. Design and implementation of frost: Digital forensic tools for the openstack cloud computing platform.

[19] Asif Imran, Alim Ul Gias, and Kazi Sakib. An empirical investigation of cost-resource optimization for running real-life applications in open source cloud. In *High Performance Computing and Simulation (HPCS), 2012 International Conference on*, pages 718–723. IEEE, 2012.

[20] Brian Neil Levine and Marc Liberatore. Dex: Digital evidence provenance supporting reproducibility and comparison. *digital investigation*, 6:S48–S56, 2009.

[21] Mark Taylor, John Haggerty, David Gresty, and Robert Hegarty. Digital evidence in cloud computing systems. *Computer Law & Security Review*, 26(3):304–308, 2010.

[22] Kai Zeng, Kannan Govindan, Prasant Mohapatra, et al. Chaining for securing data provenance in distributed information networks. In *Military Communications Conference, 2012. MILCOM 2012. IEEE*, pages 1–6. IEEE, 2012.

[23] Amir Vahid Dastjerdi and Rajkumar Buyya. An autonomous reliability-aware negotiation strategy for cloud computing environments. In *Cluster, Cloud and Grid Computing (CCGrid), 2012 12th IEEE/ACM International Symposium on*, pages 284–291. IEEE, 2012.

[24] Olive Qing Zhang, Ryan KL Ko, Markus Kirchberg, Chun Hui Suen, Peter Jagadpramana, and Bu Sung Lee. How to track your data: Rule-based data provenance tracing algorithms. In *Trust, Security and Privacy in Computing and Communications (TrustCom), 2012 IEEE 11th International Conference on*, pages 1429–1437. IEEE, 2012.

[25] Nwokedi Idika, Mayank Varia, and Harry Phan. The probabilistic provenance graph. pages 83–102, 2013.

[26] Junjie Yao, Bin Cui, Zijun Xue, and Qingyun Liu. Provenance-based indexing support in micro-blog platforms. In *Data Engineering (ICDE), 2012 IEEE 28th International Conference on*, pages 558–569. IEEE, 2012.

[27] Ryan KL Ko, Peter Jagadpramana, and Bu Sung Lee. Flogger: A file-centric logger for monitoring file access and transfers within cloud computing environments. In *Trust, Security and Privacy in Computing and Communications (TrustCom), 2011 IEEE 10th International Conference on*, pages 765–771. IEEE, 2011.

[28] Anderson Marinho, Leonardo Murta, Cláudia Werner, Vanessa Braganholo, Sérgio Manuel Serra da Cruz, Eduardo Ogasawara, and Marta Mattoso. Provmanager: a provenance management system for scientific workflows. *Concurrency and Computation: Practice and Experience*, 24(13):1513–1530, 2012.

[29] Zhiwei Yu, Chaokun Wang, Clark Thomborson, Jianmin Wang, Shiguo Lian, and Athanasios V Vasilakos. A novel watermarking method for software protection in the cloud. *Software: Practice and Experience*, 42(4):409–430, 2012.

[30] Martin Salfer, Sven Wohlgemuth, Sebastian Schrittwieser, Bernhard Bauer, and Isao Echizen. Data provenance with watermarks for usage control monitors at disaster recovery. In *Internet of Things (iThings/CPSCom), 2011 International Conference on and 4th International Conference on Cyber, Physical and Social Computing*, pages 514–519. IEEE, 2011.

[31] James Cheney. A formal framework for provenance security. In *Computer Security Foundations Symposium (CSF), 2011 IEEE 24th*, pages 281–293. IEEE, 2011.

[32] Jinhui Yao, Shiping Chen, Chen Wang, David Levy, and John Zic. Accountability as a service for the cloud. In *Services Computing (SCC), 2010 IEEE International Conference on*, pages 81–88. IEEE, 2010.

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

[34] Huan-Chung Li, Po-Huei Liang, Jiann-Min Yang, and Shiang-Jiun Chen. Analysis on cloud-based security vulnerability assessment. In *e-Business Engineering (ICEBE), 2010 IEEE 7th International Conference on*, pages 490–494. IEEE, 2010.

[35] Kiran-Kumar Muniswamy-Reddy, Peter Macko, and Margo I Seltzer. Provenance for the cloud. In *FAST*, volume 10, pages 15–14, 2010.

[36] Jinwei Hu, Yan Zhang, Ruixuan Li, and Zhengding Lu. Managing authorization provenance: A modal logic based approach. In *Tools with Artificial Intelligence, 2009. ICTAI'09. 21st International Conference on*, pages 621–624. IEEE, 2009.

[37] Abha Moitra, Bruce Barnett, Andrew Crapo, and Stephen J Dill. Data provenance architecture to support information assurance in a multi-level secure environment. In *Military Communications Conference, 2009. MILCOM 2009. IEEE*, pages 1–7. IEEE, 2009.

[38] Kai Hwang, Sameer Kulkareni, and Yue Hu. Cloud security with virtualized defense and reputation-based trust mangement. In *Dependable, Autonomic and Secure Computing, 2009. DASC'09. Eighth IEEE International Conference on*, pages 717–722. IEEE, 2009.

[39] Soila Kavulya, Jiaqi Tan, Rajeev Gandhi, and Priya Narasimhan. An analysis of traces from a production mapreduce cluster. In *Cluster, Cloud and Grid Computing (CCGrid), 2010 10th IEEE/ACM International Conference on*, pages 94–103. IEEE, 2010.

[40] Jiaqi Tan, Xinghao Pan, Soila Kavulya, Rajeev Gandhi, and Priya Narasimhan. Mochi: visual log-analysis based tools for debugging hadoop. In *Proceedings of the 2009 conference on Hot topics in cloud computing*, page 18. USENIX Association, 2009.

[41] Juliana Freire, David Koop, Emanuele Santos, and Cláudio T Silva. Provenance for computational tasks: A survey. *Computing in Science & Engineering*, 10(3):11–21, 2008.

[42] James Frew, Dominic Metzger, and Peter Slaughter. Automatic capture and reconstruction of computational provenance. *Concurrency and Computation: Practice and Experience*, 20(5):485–496, 2008.

[43] Xuxian Jiang, Aaron Walters, Dongyan Xu, Eugene H Spafford, Florian Buchholz, and Yi-Min Wang. Provenance-aware tracing ofworm break-in and contaminations: A process coloring approach. In *Distributed Computing Systems, 2006. ICDCS 2006. 26th IEEE International Conference on*, pages 38–38. IEEE, 2006.

[44] Siani Pearson and Azzedine Benameur. Privacy, security and trust issues arising from cloud computing. In *Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on*, pages 693–702. IEEE, 2010.

[45] Wenchao Zhou, Qiong Fei, Shengzhi Sun, Tao Tao, Andreas Haeberlen, Zachary Ives, Boon Thau Loo, and Micah Sherr. Nettrails: a declarative platform for maintaining

and querying provenance in distributed systems. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of data*, pages 1323–1326. ACM, 2011.

[46] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, et al. A view of cloud computing. *Communications of the ACM*, 53(4):50–58, 2010.

[47] Ryan KL Ko, Bu Sung Lee, and Siani Pearson. Towards achieving accountability, auditability and trust in cloud computing. In *Advances in Computing and Communications*, pages 432–444. Springer, 2011.

[48] Rostyslav Slipetskyy. *Security issues in OpenStack*. PhD thesis, Masters thesis, Norwegian University of Science and Technology, 2011.

[49] Shams Zawoad, Amit Kumar Dutta, and Ragib Hasan. Seclaas: secure logging-as-a-service for cloud forensics. In *Proceedings of the 8th ACM SIGSAC symposium on Information, computer and communications security*, pages 219–230. ACM, 2013.

[50] Dimitrios Zissis and Dimitrios Lekkas. Addressing cloud computing security issues. *Future Generation Computer Systems*, 28(3):583–592, 2012.