PRACTICAL BAYES APPROACH FOR DETECTION OF SOFTWARE DEFECTS IN CLOUD SERVICES

Nethaji V[1], *Chandrasekar C*[2]

ABSTRACT

Now a days Cloud computing plays a vital role in the IT enterprise. Statistical Process Control cloud charts sense routine variances and their root causes are identified based on the differential profiling strategy. Most of the manual overhead incurred in detecting the software variances and the analysis time are reduced to a larger extent but detailed analysis of profiling data are not performed in most of the cases. At the same time, Trusted Computing Base (TCB) of a computing node does not achieve the scalability measure. This work, a Practical Bayes approach studies the problem of detecting software variances and ensures scalability by comparing information at the current time to historical data. GenProg uses an extensive structure of genetic programming to develop a program variant that retains essential functionality but it is not vulnerable to a known deficiency in cloud. The existing software testing suite identifies program defects in cloud environment. Delta debugging and Structural differencing algorithms minimize the dissimilarity among variant and the original program in terms of minimum repair. Subsequently, Defect Localization based on Band (DLB) mechanism is introduced to overcome the defects and rank the different acceptable patches.

Keywords: Cloud Environment, Variances, Practical Bayes approach, Gaussian mixture, Trusted Computing Base, Genprog.

I. INTRODUCTION

Cloud Computing is being altered and distorted to a new model consisting of services that are commoditized and delivered in a fashion analogous to conventional utilities. In such a model, customers access services based on their necessities without knowing from where the services are hosted or how they are distributed. Cloud computing denotes the infrastructure as a Cloud from which commerce and clients are experienced and proficient to access applications from anywhere in the world using on demand techniques. The Cloud based Computing Service Model is based on three primary factors such as Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS). All IT functions with applications, networking, security, storage space and software are developed for users to work in a service, based on the client server model.

An innovative remote attestation framework called DRAFT as illustrated in [7] for efficient measuring of target system based on an information flow-based integrity model. The high integrity processes of a system are first measured and established, and these processes are then confined from accesses initiated by low integrity processes. An efficient cryptographic protocol as shown

[1]Research Scholar, Department of Computer Science, Karpagam University, Coimbatore - 641 120 Tamil Nadu, India Email: nethaj.babu@gmail.com. Mobile: +91-9962070800

[2]Assistant Professor, Computer Science Department, Periyar University, Salem, 636011, India Email: ccsekar@gmail.com. Mobile: +91-9994599967

in [12] that enforces keystroke integrity by utilizing on-chip Trusted Computing Platform (TCP) prevents the counterfeit of fake key events by malware under reasonable assumptions. A reasonable assumption is difficult in accessing a host's kernel, and the facility to build application-level fine-grained detection solutions.

Collaborative provable data possession scheme as shown in [3] uses the techniques of Homomorphic demonstrable responses and hash index hierarchy. Collaborative fails to expand more effective and practical CPDP constructions. First performance of CPDP scheme, especially for large files, is seriously affected by the bilinear mapping operations because of high complexity. Cooperative PDP (CPDP) scheme proves the security based on multi proving zero-knowledge proof system in [10], which assure unity bit but it is affected by the bilinear mapping operations due to its high complexity. Additionally, articulate performance optimization mechanisms for CPDP scheme present an efficient method for identifying the parameter values to reduce the cost involved during computation of clients and storage service providers.

Hierarchical Attribute Set Based Encryption (HASBE) extended cipher text-policy Attribute-Set-Based Encryption (ASBE) with a hierarchical structure of users. The ASBE scheme as shown in [15] not only attains scalability due to its hierarchical arrangement, but also inherits elasticity and fine-grained accesses manage in supporting compound attributes of ASBE. ASBE efficiently share confidential data on cloud servers using Hierarchical Identity Based Encryption (HIBE) system and the Cipher Text-Policy Attribute-Based Encryption (CP-ABE) system, and finally providing performance expressivity trade off as described in [12].

The Secure cloud storage system as depicted in [10] supports privacy-preserving public auditing which performs audits for multiple users concurrently and proficiently. Public auditability for cloud storage is of serious consequence so that users resort to a Third Party Auditor (TPA) as shown in [13] check the integrity of outsourced data. To securely establish an efficient TPA, the auditing process brings in novel vulnerabilities towards user data privacy. As described in [5] bread and butter of data forensics and post investigation in cloud computing is characterized by providing the information privacy on sensitive documents

Nefeli, a virtual infrastructure gateway as demonstrated in [8] provides deployment hints on the probable mapping of VM to physical nodes. The existence of possible performance bottlenecks, and the existence of underlying hardware features. Nefeli investigate of alternative constraint satisfaction approaches to address scalability issues present in large infrastructures and failed to offer deployment hints that efficiently handle the deployment of virtual infrastructures in the background of real large cloud installations. SBSE for the cloud as formulated in [13] challenges by way of addressing search based software engineering. Cloud providers share analogous goals in reducing resource usage, but they are less focused on upholding their service level agreements on intrusions.

An effective fault localization mechanism would return a root cause using the apprehensive list program elements. Although existing method with fault localization is effective only on some of the cases, regrettably, for many other cases, fault localization methods are not effectual sufficient. GenProg as shown in [15] is an automated method for repairing defects in off-the-shelf, legacy programs without official condition, program annotations, or particular coding practices. Structural differencing algorithms and delta debugging decrease the difference between this variant and the unique program to a least repair. Root causes are often listed low in the record of most distrustful program elements. The unreliability of fault localization tools potentially motive many developers to distrust fault localization methods.

In Practical Bayes (PB) approach, two component Gaussian mixtures are used to perform deviations. PB monitors the massive number of cells which is useful in streaming scenarios with Bayes per section error rate procedure. A novel feature of PB approach is the capability to restrain deviations that simplify the consequence of sharp changes in the marginal distributions. The contribution of Practical Bayes approach is to present a PB framework to detect software variances in imbalanced classified data streams with potentially large number of cells. PB framework performs multiple testing using a hierarchical Bayesian model and suppresses redundant alerts caused due to changes in the marginal distributions.

The structure of paper is as follows. Section 1, describes the Practical Bayes Theoretical Framework. Section 2, describes the Defect Localization Mechanism. Section 3 describes the PB Experimental Approach with Parametric Factors. Section 4 describes the Illustration analysis the result through table and graph values and section 5 describes Conclusion.

II. Practical Bayes Theoretical Framework

Practical Bayes aims to detect the software anomalous behaviour by comparing data in the current block based on historic data. However, PB approach is interested in detecting software anomalous patterns rather than detecting abnormal software records. The design of PB approach is centred on the concept of monitoring statistical measures which are computed for combinations of definite attributes in the database. Definite attribute combinations give rise to multi software's testing at each interval. In order to achieve multi software testing, Bayes per section error rate is evaluated on each cell, where each dimension corresponds to the levels of a categorical variable. The framework of PB is shown in Fig 1.

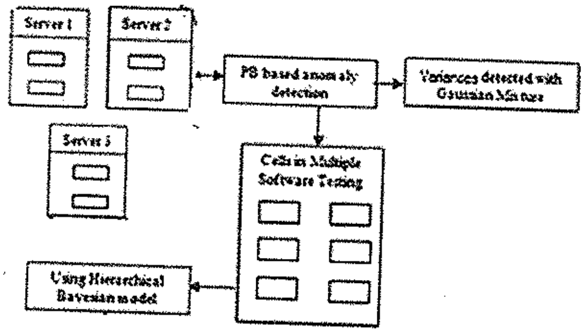
Server 1
Server 2
Server 3
PB based anomaly detection
Variances detected with Gaussian Mixture
Cells in Multiple Software Testing
Using Hierarchical Bayesian model

Figure1 : Architecture Diagram of Practical Bayes approach

As shown in Fig 1, PB classifies the imbalanced data streams. Imbalanced data stream gets computed from call logs that are added to the current database on a daily basis. Software variances which are direct consequences of changes in a small number of margins are detected with Gaussian mixture.

For ease of evaluation, the PB approach proceeds with the assumptions that the multidimensional software tests consist of two categorical variables with 'A' and 'B' levels respectively. 'A' and 'B' levels note the generalization of higher dimensions software variances. In practice, PB takes the suffix of the 'A' and 'B' levels for the first and second categorical variables respectively at time 't'. Let X_{ABT}

denote the observed value to follow a Gaussian distribution. Often, a certain level of transformation is required for the original data to ensure approximation of true value.

In order to ensure approximate software normality, the counts are observed with the help of a square root transformation. In general, PB square root transformation is denoted as

$$(x + l)^{q}\text{-}1)/\text{q}) \qquad (1)$$

In (1), 'l' and 'q' are chosen to 'stabilize' the variance and depends on the mean recommended in software anomaly detection. Moreover, 'q' is constrained to lie between 0 and 1, and qà0 implies a log transformation while detecting the software variances. In fact, the values of these parameters are chosen with a reasonable value with the help of the initial training data. For time interval 't' PB approach detects the software variances after regulating the changes with marginal means. The process of PB approach starts with the formal definition of software anomaly followed by the mechanism Practical Bayes approach explained with the help of an algorithm.

A. Formulation of PB Software Anomaly

Consider a 2*2 table, the levels of the row feature being I, J and the levels of the column feature being i, j respectively. PB denotes the 4 cell entries corresponding to (Xx, Xy, Yx, YY) by a vector of length 4.

$$\begin{matrix} Xx & Xy \\ Yx & YY \end{matrix} \qquad (2)$$

Two values are measured in PB approach to analyse the scalability issue, namely, the expected values and the observed values. The deviations after adjusting the changes in the row and column means are (0, 0, 0, 0), ensuring that it produces no software variances in cloud environment. The significant values in the PB approach and the non-adjusted changes is described to be a drop in the first row mean and a rise in the second row mean. Hence, non-adjusted cell modification contains redundant information which results in a situation where the adjusting for margins is desirable.

Let C_{t-1} denote the current historical information up to time t-1. Deviations at time t are detected by comparing the observed values with the corresponding posterior analytical Practical Bayes distributions. The PB expected distribution of data at time t is based on current historic data until t-1. Gaussian mixture with Practical Bayes mean

$$\mu_{ABT} = P(X_{ABT}|C_{t-1}) \qquad (3)$$

$$\sigma^2_{ABT} = Var(X_{ABT}|C_{t-1}) \qquad (4)$$

and variances is computed as given below. μ_{ABT} represent the mean of PB approach and σ^2_{ABT} represent the variance factor of the PB approach. X_{ABT} denote the observed value which is assumed to follow a Gaussian distribution mixture. The Gaussian denotes current historical information up to time t-1 for detailed analysis of profiling data in cloud environment.

The central idea of the Hierarchical Bavesian model is to classify the imbalanced data streams. $H_{mixture}$ assumes Δ_{ABT} which are random samples from Gaussian mixture distribution at time 't', GM_t. The form of is known but depends on the densily rate parameter. A Bayesian hierarchical model is one that is written with software modularity. It is often useful to think of the analysis of data streams using PB inside-unit analysis, and another model for the across-unit analysis. The inside-unit model describes the behaviour of individual respondents over run time, while the across-unit analysis describes the density and complexity of the units. The two models, inside-unit and across-unit b combine to form the hierarchical model, and Practical Bayes algorithm is used to integrate the pieces mutually and report for all the uncertainty and variances.

A Practical Bayes approach makes inference about by using approximation of the hyper parameters in order to perform the deviations. The software inference is obtained by numerically integrating with respect to the posterior of using an adaptive Gaussian mixture quadrature. PB Gaussian posterior distribution of depends directly on and indirectly on the other 's through the posterior of the hyper parameters in order to reduce the complexity issue. Generally, such adaptation of strength makes the posterior means of regress toward each other and automatically builds penalty for conducting multiple software tests.

III. Defect Localization Mechanism

The goal of Defect Localization based on Band (DLB) mechanism is to build a model that predict defect in an effective way. To realize DLB, illustrated in Fig2, Amazon EC2 dataset are taken for the defect localization in cloud environment. Information from Amazon EC2 dataset is leveraged to predict defect localization model with different set of program execution traces with the defect being localized based on the band. Band in DLB mechanism depends on the spectra that contains ordered list of program elements. The ordered list of program elements in DLB mechanism are sorted based on the likelihood. Defect Localization based on Band extracts features in cloud that are potentially associated to perform effective ranking in cloud environment.

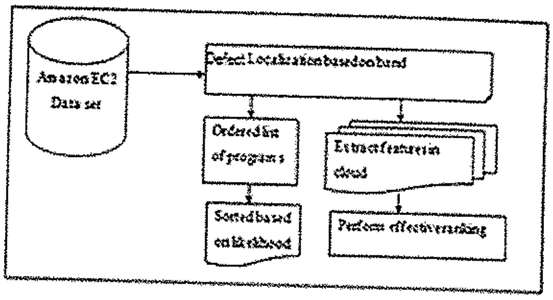
Amazon EC2 Data set
Defect Localization based on band
Ordered list of programs
Sorted based on likelihood
Extract features in cloud
Perform effective ranking

Figure 2. Overall Architecture Diagram of DLB Mechanism

In the special case, as depicted in fig 2, where all program elements are given the same dishonest score values, there is a very low likelihood that the defect localization model is effective for those execution traces. Defect localization comprises two phases namely instruction phase and operation phase. In fact, the adjusted software version detects changes in interactions among the levels of categorical variables which are the focus of several applications. Also, in higher dimensions PB approach adjust for higher order margins, which is routine in PB approach framework. For instance, adjusting two-way margins in a three dimensional form detects changes in third order interactions also.

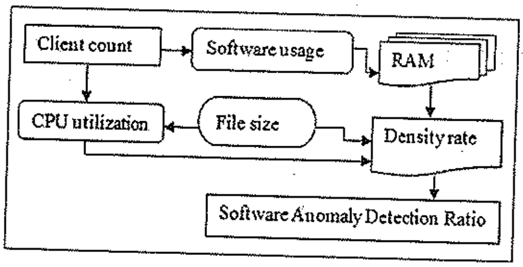

Figure 3. Model for Software Anomaly using PB

Fig 3 describes the detection of software variances based on the Practical Bayesian approach. The client count in cloud environment identifies the CPU utilization and software usage factor. The software stored and RAM usage is resolute. The basic model of PB approach is analyzed and finally density rate and software anomaly detection ratios are evaluated. The cell penalty increases with software predictive variance. Also, the general penalty of the procedure at time 't' depends on the hyper parameters which are predictable from data.

Input: Cloud servers with software's

Output: Detect Software variances in cloud environment

Start

Step 1: Produce μ_i for Gaussian distribution 'G'

Step 2: Each data from training data and fitted with a normal Gaussian distribution

Step 3: Mean, Median and variance of cells in multiple software testing is generated

Step 4: For each i, simulate H $(\mu_{ABT}, \sigma^2_{ABT})$

Step 5: At time't', PB selects the data streams, and then software variances are generated.

Step 6: Find the software variances at time't'

Step 7: Hierarchical Bayesian Model classifies the imbalanced data streams till the last row and column. End

In principle, PB model provides an estimate of posterior Gaussian predictive means and variances to obtain $\mu_{ABT}, and\ \sigma^2_{ABT}$. However, elaborating on appropriate Gaussian mixture on PB approach has been chosen and the software is trained analytically by the user. Also, to be useful in data streaming scenarios, the PB model is easily adapted to new data. Gaussian mixture effectively captures c_t and detects the software variances in cloud environment. Then, the posterior predictive mean μ_{ABT} is the sample mean and the posterior predictive variance σ^2_{ABT} is replaced by its estimator s^2_{ABT} for effective variance on each x_{ABT}. In order to adjust effects, a separate Gaussian mixture is maintained for each looping.

IV. PB EXPERIMENTAL APPROACH WITH PARAMETRIC FACTORS

Performance metric for evaluation of PB approach is measured in terms of runtime, software anomaly detection ratio, CPU utilization and density rate. Runtime factor is defined as the amount of time consumed to perform the software variances detection, measured in terms of seconds. Scalability factor measures the quality of services provided using the Practical Bayes approach, measured in terms of percentage. The characteristic of a PB system is that it describes its capability to cope and perform an increased detection service.

Software anomaly detection ratio in PB is measured as the amount of time consumed to perform the operations on cloud using Gaussian mixture to detect the variances whereas the CPU utilization is amount of CPU cycles undergone to perform the variances detection operation, measured in terms of Kilobits per second. Finally, the density rate is the average speed of detecting the variances, measured in terms of percentage.

V. ILLUSTRATION

Practical Bayes (PB) approach in cloud environment is compared against the existing Statistical Process Control (SPC) framework and Trusted Computing Base (TCB) with Open Stack prototype on Amazon EC2 dataset. The experimental value through table and graph describes the software variances detection parametric factors on cloud environment.

Table 1. Tabulation of Runtime

No. of users	Runtime (sec)		
	SPC Framework	TCB	PB approach
5	95	81	77
10	135	120	107
15	217	202	186
20	242	221	211
25	426	386	346
30	536	481	446
35	836	797	727

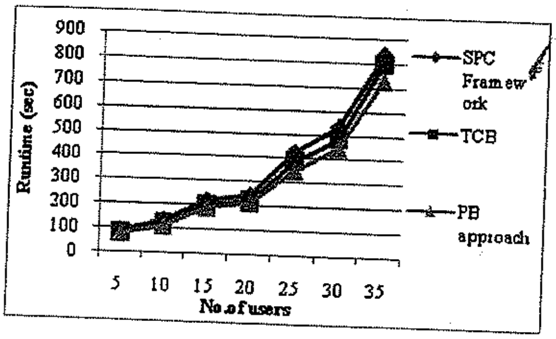

Figure 4. Measure of Runtime

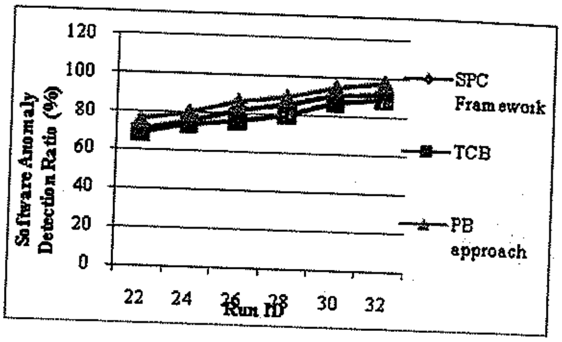

Figure 5. Software Anomaly Detection Ratio Measure

Table 1 and fig 4 illustrate the runtime based on the user count. The inside unit model in PB approach describes the behaviour of individual respondents over run time and reduces the percentage by 13%- 19% when compared with SPC Framework [1]. The models combine to form the hierarchical model, and Practical Bayes algorithm is used to integrate the pieces mutually and report for all the uncertainty and variances within 5 – 10 % limited runtime when compared with TCB [2].

Table 2 and fig 5 illustrate the software anomaly detection ratio is measured based on the Run Id count. PB approach improves the detection ration by 3 – 8 % when compared with SPC Framework [1] because square root transformation is adequate, for proportions arcsine which detect the software anomaly. In PB approach, software track changes in the margins separately using simple process control techniques and run both adjusted and unadjusted versions and results in 7 – 17 %improved detection ratio when compared with TCB [2].

Table 2. Tabulation for Anomaly Detection Ratio

Run Id	Software Anomaly Detection Ratio (Success %)		
	SPC Framework	TCB	PB approach
20	65	62	72
22	70	65	74
24	75	72	82
26	81	75	85
28	85	80	87
30	90	85	97
32	92	87	97

Table 3. Tabulation for CPU utilization

File Size (KB)	CPU utilization (Kbps)		
	SPC Framework	TCB	PB approach
35	6662	5651	4640
62	6735	6730	5720
90	7535	6526	5821
124	7852	6384	5920
189	8165	7155	6147
225	8232	7515	6596
387	9550	8528	7523

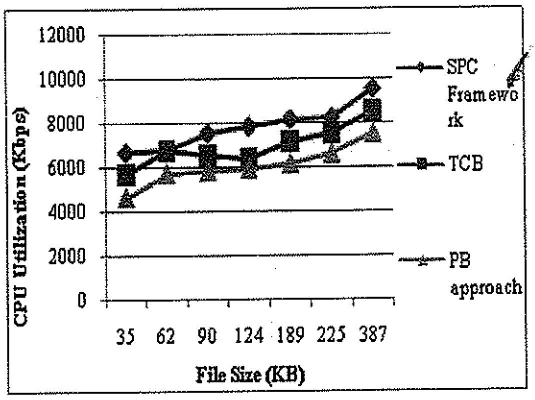

Figure 6. Performance of CPU utilization

Table 3 and Figure 6 illustrate the CPU utilization based on the File size. The size of file is measured in terms of Kilobytes (KB). Practical Bayes approach used in process control estimate Δ_{ABT} with δ_{ABT} and declare the ab^{th} cell of a software anomaly to reduce the CPU utilization. Deviations at time t are detected by comparing the observed values with the corresponding posterior analytical Practical Bayes distributions, so that it results in minimum CPU utilization of 15 – 32 % when compared to SPC Framework [1] and reduces from 8-18% when compared with TCB [2].

Table 4. Tabulation for Density Rate

Performance Counter	Density Rate (%)		
	SPC Framework	TCB	PB approach
10	60	64	67
20	67	68	72
30	77	78	82
40	75	80	87
50	76	82	86
60	77	82	87
70	84	88	94

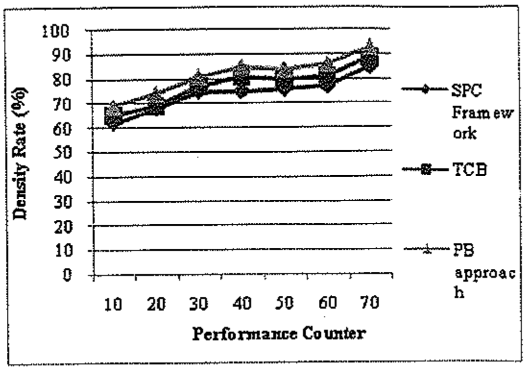

Figure 7. Density Rate Measure

Table 4 and Figure 7 illustrate the density rate which is measured based on the performance counter. The performance count ranges from 10, 20, 30…up to 70. The density rate is improved by imbalanced classified data stream from call logs that are added to the current database. When monitoring cells for deviations, the PB approach regulates the marginal statistics and increases density rate by 8 – 14 % when compared with SPC Framework [1]. Software variances which are direct consequences of changes in a small number of margins are detected with Gaussian mixture, and improve the density rate from 4 – 8 % when compared to the TCB [2].

VI. Conclusion

In this paper, a software variances detection model called the Practical Bayes approach is presented which aims to detect the software variances in massive imbalanced classified data streams. The values of these features from an instruction set of defect build a discriminative model using machine learning. DLB extracts the features in cloud that are potentially associated for effective ranking. The PB framework then reduces redundancy by adjusting marginal changes and solves the multiple software testing problems using hierarchical Bayesian model within a decision theoretic framework. PB approach proves the superiority of $H_{mixture}$ through simulation using the two component Gaussian mixture for deviations in cloud environment. Furthermore, the PB approach works on combining adjusted and unadjusted to automatically produce software variances detection. The Fault detect mechanism is then used as an ordered list of program elements sorted based on their likelihood. The techniques normally change program runtime states methodically to localize faulty program elements. It focuses on fault localization tools that compare and correct and faulty executions. Experimental result attains the 9.256% minimal runtime and CPU utilization. PB also improves the density rate, scalability, and software anomaly detection ratio on Amazon EC2 dataset.

Acknowledgement

I am greatly indebted to my guide, DR.C.Chandrasekar M.C.A., Ph.D, Associate Professor in Department of Computer Science, Periyar University, for his unstinted encouragement and guidance for the successful completion of this manuscript.

I have to thank my parents, wife, son and friends for their encouragement and timely suggestions to successfully prepare this valuable manuscript.

References

[1] Donghun Lee., Sang K. Cha., and Arthur H. Lee., *"A performance anomaly detection and analysis framework for DBMS development,"* IEEE Transactions on Knowledge and Data Engineering, vol. 24, no. 8, August 2012

[2] Imad M. Abbadi., and Anbang Ruan., *"towards trustworthy resource scheduling in clouds,"* IEEE Transactions on Information Forensics and Security, vol. 8, no. 6, June 2013

[3] Flavio Lombardi., RobertoDiPietro., *"Secure virtualization for cloud computing,"* Journal of Network and Computer Applications, Elsevier journal., 2010

[4] A.S.Syed Navaz., V.Sangeetha., C.Prabhadevi., *"Entropy based anomaly detection system to prevent DDoS attacks in cloud,"* International Journal of Computer Applications (0975 – 8887) Volume 62– No.15, January 2013

[5] Rongxing Lu., Xiaodong Lin., Xiaohui Liang., and Xuemin (Sherman) Shen., *"Secure provenance: The essential of bread and butter of data forensics in cloud computing,"* ACM journal., 2010

[6] Vivek Nallur., Rami Bahsoon., *"A decentralized self-adaptation mechanism for service-based applications in the cloud,"* IEEE Transactions on Software Engineering, 2012

[7] Wenjuan Xu, Xinwen Zhang., Hongxin Hu., Gail-Joon Ahn., and Jean-Pierre Seifert., *"Remote attestation with domain-based integrity model and policy analysis,"* Transactions on Dependable and Secure Computing, vol. 9, no. 3, IEEE, 2012

[8] Konstantinos Tsakalozos., Mema Roussopoulos., and Alex Delis., *"Hint-based execution of workloads in clouds with nefeli,"* IEEE Transactions on Parallel and Distributed Systems, vol. 24, no. 7, July 2013

[9] Alexandru Iosup., Simon Ostermann,Nezih Yigitbasi., Radu Prodan., Thomas Fahringer., and Dick Epema., *"Performance analysis of cloud computing services for many-tasks scientific computing,"* IEEE TPDS, Many-Task Computing, Nov 2010.

[10] Mohamed Nabeel., Elisa Bertino., *"Privacy-preserving fine-grained access control in public clouds,"* IEEE Computer Society Technical Committee on Data Engineering, 2012

[11] Kui Xu., Huijun Xiong, Chehai Wu., Deian Stefan., and Danfeng Yao., *"Data-provenance verification for secure hosts,"* IEEE Transactions on Dependable and Secure Computing, vol. 9, no. 2, March/April 2012

[12] Qian Wang., Cong Wang., Kui Ren., Wenjing Lou., and Jin Li., *"Enabling public auditability and data dynamics for storage security in cloud computing,"* IEEE Transactions on Parallel and Distributed Systems, vol. 22, no. 5, May 2011

[13] Dimitrios Zissis., Dimitrios Lekkas., *"Addressing cloud computing security issues,"* Future Generation Computer Systems., Elsevier journal., 2012

[14] Yan Zhu., Shanbiao Wang., Hongxin Hu Gail-Joon *"Secure Collaborative Integrity Verification for Hybrid Cloud Environments,"* World Scientific Publishing Company., International Journal of Cooperative Information Systems, DOI: 10.1142/S0218843012410018., Vol. 21, No. 3 (2012)

[15] Claire Le Goues, ThanhVu Nguyen, Stephanie Forrest., and Westley Weimer, *"GenProg: A Generic Method for Automatic Software Repair,"* IEEE Transactions on Software Engineering, vol.38, no.1, January/February 2012

AUTHOR'S BIOGRAPHY

V.Nethaji, Tamilnadu, India, 01 November 1981. I received Bachelor of Science (B.Sc) in Computer Technology (Regular) from K.S.Rangasamy College of Technology, Tiruchengode, Tamilnadu, India in 2003. I completed Master of Computer Applications (M.C.A) in Computer Application (Regular) from Periyar University, Salem, Tamilnadu, India in 2006 and Master of Philosophy (M.Phil.) in Computer Science (Distance) from Periyar University, Salem, Tamilnadu, India in 2008.

I have 7 years' experience in Software Testing. Currently I am working in American Megatrends India P Ltd, Chennai as Test Engineer. Previously I have worked in Computer Sciences Corporation as Associate Test Engineer.

Dr.C.Chandrasekar, Tamilnadu, India, 16 May 1972. He received Master of Computer Applications (MCA) from Madurai Kamarajar University, Madurai, Tamilnadu, India and Doctor of Philosophy (PhD) from Periyar University, Salem, Tamilnadu, India.

He has 17 years of experience in both Teaching and research. Currently he is working as Associate Professor in Department of Computer Science from Periyar University, Tamilnadu, India. Previously he worked as Assistant Professor in K.S.Rangasamy College of Technology.

[1] Chandrasekar. C and Venkatesan. K., "Introduction to Information Technology", Serve the People Publications, 2010.

[2] Dr. C.Chandrasekar/S.Adaekalavan "Towards New Estimating Incremental Dimensional Algorithm (EIDA)" Journal of Theoretical and Applied Information Technology, Volume 51, Number 2, May 2013, pp. 222-227.

[3] Dr.C.Chandrasekar/ S.Sivakumar "Lung Nodule Detection Using Fuzzy Clustering and Support Vector Machines" International Journal of Engineering and Technology (IJET), Vol 5 No 1 Feb-Mar 2013, pp. 179-185.