International Journal of Computational Intelligence Systems, Vol. 5, No. 5 (September, 2012), 924-932

Trusted Bytecode Virtual Machine Module: A Novel Method for Dynamic Remote Attestation in Cloud Computing

Songzhu Mei, Zhiying Wang, Yong Cheng, Jiangchun Ren, Jiangjiang Wu, Jie Zhou
School of Computer, National University of Defense Technology, 410073, Changsha, Hunan, P.R.China
{sz.mei, zywang, ycheng, jcren, jiangwu, jiezhou}@nudt.edu.cn
www.nudt.edu.cn

Received 30 November 2011
Accepted 19 June 2012

Abstract

Cloud computing bring a tremendous complexity to information security. Remote attestation can be used to establish trust relationship in cloud. TBVMM is designed to extend the existing chain of trust into the software layers to support dynamic remote attestation for cloud computing. TBVMM uses Bayesian network and Kalman filter to solve the dynamicity of the trusted relationship. It is proposed to fill the trust gap between the infrastructure and upper software stacks.

Keywords: bytecode virtual machine, trusted computing, cloud computing, remote attestation, Bayesian network.

1. Introduction

Cloud Computing has generally emerged as one of the most influential technologies in both the IT industry and academia. Cloud computing, as the Salesforce's definition, is a friendlier mode for business operation. It is rapidly revolutionizing the way IT resources are managed and utilized[1]. Cloud computing endows itself a lot of outstanding characteristics, such as on-demand self-service, ubiquitous network access, resource pooling, rapid elasticity and measured service[2].

Cloud computing, as a novel computing resource organizing methods, can provide scalable, flexible and unlimited storage service. The most significant change it brings to the users is that the cloud service providers can provide the consumers a high-performance, high-efficiency and pay-as-you-go computing capability. Consumers can not only use the cloud service as a data storage space, but also a platform on which to deploy and manage their business processes.

In Ref. 2, the National Institute of Standards and Technology (NIST) has proposed that cloud model is composed of three service models.

- Software as a Service (SaaS). The capability provided to the consumer is to use the provider's applications running on a cloud infrastructure. The consumer does not manage or control the underlying cloud infrastructure.
- Platform as a Service (PaaS). The capability provided to the consumer is to deploy consumer-created applications using programming languages and tools supported by the provider. The consumer only has control over the consumer-deployed applications and the configurations of the hosting environment.
- Infrastructure as a Service (IaaS). The capability provided to the consumer is to provision fundamental computing resources where the consumer is able to deploy and run arbitrary software including operating systems and applications. The consumer has control over operating systems, storage, and deployed applications.

Companies can choose any of above models to deploy their applications according to the functionally and security concerns. Obviously, the IaaS model can provide most security assurance to the consumer, because it is more convenient for the users to enforce their own security mechanisms and policies. Ref. 3 and 4 has been designed to enhance the business processes' security in the IaaS environment and users can trust cloud-based applications built on it. However, when using the former two service models, consumers can hardly control the applications completely. Recently, most trust relationship established in cloud environment is social trust[5]. Social trust is a trust that arises between two entities based upon social relationships. For example, a consumer can trust Google App Engine (GAE) for the reputation of Google Corporation. But no one can guarantee that there is no design flaw in GAE, so there must establish some kinds of technical trust to provide secure assurance in a determinate way.

Trusted Computing Group (TCG), proposed a set of hardware and software technologies to enable the construction of trusted platforms[6]. In particular, the TCG proposed a standard for the design of the trusted platform module (TPM) chip. TPM can be used as a reliable root of trust for measurement (RTM), root of trust for reporting (RTR), and root of trust for storage[7] (RTS). With the help of TPM, a computing platform can build store and report a trust chain from the lower BIOS to upper OS-kernel. It endows remote party the ability to perform remote attestation[6]. A remote party can easily attest the platform's configuration of hardware and software by checking the value of platform configuration register (PCR) in the TPM.

Nowadays, with the rapid development of the cloud computing, a lot of technologies have been proposed to fulfill the need for the trust in the cloud environment. In Refs. 3-5, different kinds of trusted platform have been designed to assure the trust relationship between the consumers and providers. But most of these technologies are applied to the IaaS model, and they can have little benefits on the SaaS and PaaS model. PaaS and SaaS models place several software layers on IaaS. These software may include development tool chain, runtime libraries, and abundant applications, and they bring the system more complexity. It will be a hard job to measure all the applications' binary image, not to say their runtime states. This problem will be amplified when these two models introduce Java Virtual Machine (JVM) or .Net common language runtime (CLR) executive engine as their runtime platforms. The actual application is the bytecode for these virtual machines. The underlying OS will simply handle the application as an input file of the executive engines. Although in Ref. 8, Biba has announced that when considering the application's integrity, we cannot ignore the integrity of the application's runtime data. Sailor et al. point that it is a mission impossible for an attestation system to do so many jobs at the same time[9].

In this paper, we propose the Trusted Bytecode Virtual Machine Module (TBVMM). It is a well-designed software module that leveraging the advances of trusted computing technologies to support the remote attestation of high-level program properties and behaviors. Providers can embed TBVMM into BVM as the trust root. In this situation it extends the trust chain to the runtime platform for the applications. TBVMM has three significant advantages:

- User-involved trust management. SLA clarifies the functional constraints in a formal way. According to the SLA, TBVMM will configure the policy decision point (PDP) and the policy enforcement point (PEP). PDP and PEP will decide and enforce the security policy to protect users' applications and data.
- Virtual memory shielding. To enhance the security of the BVM and its applications, BVM should encrypt its built-in virtual memory. With the supports of TBVMM, BVM can communicate with the underlying TPM and maintain the memory-shield key in the TPM to avoid the risk of key leakage.
- Dynamic remote attestation. TBVMM will monitor the states-transition of each component in the container. If the component violates the rule or functional constraint established between the consumer and provider, TBVMM will record and report the threats to the consumers and lower the trust level of the provider.

The rest of the paper is organized as follows. Section II introduces the background of bytecode virtual machine, trusted computing, remote attestation, and shows our design motivation. Then we provide the main ideas and design details of TBVMM in Section III. Section IV overviews the related work. Finally and at last Section V gives the concluding remark of the whole paper and the outlook of our work.

2. Background

2.1 *Bytecode Virtual Machine*

The BVM is a branch of high-level language virtual machine (HLL VM). It uses a set of unified well-defined bytecode as its machine instruction and emulates a simple machine to execute bytecode. BVM maintains a slice of memory space for the executing bytecode. BVM packages core system function into its own libraries and the bytecode can only access the physical machine's memory and the essential I/O devices. So native interface are introduced to let BVM applications interact with the irregular periphery devices. Native interface also bring some threats to the physical machine. An elaborated malicious BVM application could get full control over the physical machine with the native interface. So most BVMs have built-in sandboxes, a constrained executive environment without any permission to access the physical resources, unauthorized applications will be moved into the sandbox to mitigate potential harms. The architecture of a typical BVM is shown in Fig. 1.

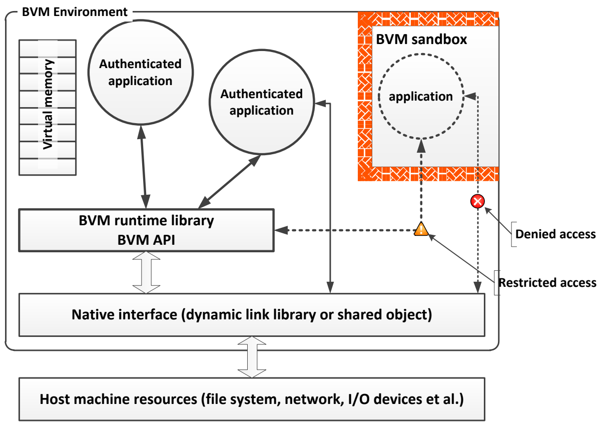
BVM Environment
Virtual memory
Authenticated application
Authenticated application
BVM sandbox
application
BVM runtime library
BVM API
Denied access
Restricted access
Native interface (dynamic link library or shared object)
Host machine resources (file system, network, I/O devices et al.)

Fig. 1. Typical BVM architecture.

BVM-like virtual machines have been widely used in the IT industry. JVM is a classical BVM implementation. We can find it almost everywhere in our human society, from server-side applications to mobile device applications. Especially in the PaaS and SaaS cloud model, JVM is adopted as the fundamental runtime environment for upper applications. GAE has used the JVM as its standard program execution platform, which is not only a support for Java language, but also other programming languages, such as Python, Ruby and PHP. Program can be compiled into JVM bytecode and then run on it without any obstacle. Salesforce.com also uses JVM to deploy its CRM applications and provision them to the consumer in a SaaS manner. In Microsoft's Azure project, CLR is also used as a unified runtime platform for C#, Java, and even C/C++ (with just-in-time mechanism) languages.

In consideration of the above reasons, BVM can be used as a junction which connects the existed trusted mechanism with various applications. So we propose TBVMM, combined with other technologies, to establish an integrated dynamic trusted executing environment.

2.2 *Trusted Computing and Remote Attestation*

Trusted computing wants to add components and mechanisms to commodity systems to bestow on them some of the properties of high-assurance closed systems like ATM. It requires three core mechanisms[7]:

- Secure boot. Make sure that the system is booted into a trusted operating system that adheres to some given security policy.
- Strong isolation. Prevent the system from being compromised after it has been booted, and to prevent applications from tampering with each other.
- Remote attestation. Certify the authenticity of software being run by a remote party.

The remote attestation is used to attest the configuration of an entity to a remote entity. This procedure is widely used to get integrity information before a client proceeds with the communication in order to use a service or receive data, e.g., digital content. This mechanism is referred as integrity reporting and can be applied in many scenarios and different applications.

The basic assumption of remote attestation is trusted server, and untrusted (even malicious) clients. Thus, even though a significant fraction of work is done at the clients, all the trust resides at the server. In conventional C/S or B/S deployment model, this assumption work well. But in the era of cloud computing, everyone can publish service for the public. These services may be well-designed. They also can have plenty of flaws and even be designed for malicious usage. The assumption for the conventional remote attestation method cannot fit cloud computing's needs well.

Another significant shortcoming is that the remote attestation is always static. It measures only the binary image of an application. This is an "all or nothing" manner. This manner is not suitable for a service-side application. Its availability cannot be assured. It is also a burden to do upgrade and patching for an application.

3. Trusted Bytecode Virtual Machine

The most significant shortcomings of traditional ways of remote attestation can be traced back to one root cause — what is desired is attestation of the behavior of software running on a remote machine, but what actually gets attested is an individual binary is being run[10]. In fact, even if the consumer clearly knows what application is being used exactly, he has no awareness about what the application really does. Use email service as an example. Suppose company D runs a popular email service, named DMail. When we use a DMail, the attester may show us that we are indeed interacting with a correct and non-tampered DMail server, and what is running in the server is absolutely the DMail service. Then we write letters and click send button. We may take for granted that our letter would have been immediately transported to the receiver without any error and leakage happen. But as a matter of fact, we have had no idea on what have been done in the DMail server. The situation would turn even worse in cloud environment. Especially the PaaS and IaaS cloud models that the consumers are allowed to publish their own services in the cloud.

We present the trusted bytecode virtual machine module (TBVMM) that provides a closed box execution environment by extending the concept of trusted computing platform to an entire PaaS and SaaS models. The TBVMM guarantees the runtime integrity and behavior compliance of a server-side application.

3.1 *The Requirements for TBVMM*

There are some essential requirements for TBVMM to fulfill the need for dynamic runtime remote attestation in cloud environment.

- Memory sealing. A batch of technologies has been developed to crack BVMs. Crackers have shown their excellent skills to break into the JVM's nutshell. The built-in virtual memory, although announced to be well-protected, is an important vulnerability for the crackers to leverage. TBVMM must have its memory protected in a cryptographic method with the underlying trusted environment.
- User-defined security policy. TBVMM shall support the users to define and enforce their security policy. TBVMM can dynamically monitor and evaluate the applications' behavior. Any violation should be recorded and reported to the user. TBVMM will also help the user to do the up-front trust-degree evaluation of an application.
- Dynamic remote attestation. TBVMM encourages application vendors to modularize their applications. Modularization can significantly reduce the states-space complexity of an application. So TBVMM can dynamically monitor the modules' states-transition more efficiently. TBVMM, in company with sandbox, can also test a service according to its vendor's formal description, e.g. WSDL* in Web Service.

3.2 *TBVMM Architecture*

Fig. 2 illustrates the architecture of TBVMM. A TBVMM at least has six core components and three derived configurable utilities. The components include:

- User Policy Analyzer (PA). This component parses the user defined security policy using XACML, and generates policy description in a formal and semantic way. The policy description will then be used to configure the derived utilities.
- Encryption Module (EM). This component interacts with the underlying TPM to maintain the memory-shielding keys. Each application has its own memory access key, so we can strongly ensure the memory protection.
- Pre-load Tester (T). This component works with sandbox. When a consumer tries to publish its application as a public service, the first-thing-first to do is to submit a description of its application. The tester will test and record its states-transition for users to evaluate its compliance with their security policy.
- Logger/Reporter (LnR). In company with underlying TPM, this component logs the application's activities securely, and can respond the users' attest requests.
- Runtime Attester (RA). This is the most important component in TBVMM. It attests the application's behavior with the help of PDP, PEP, and access monitor. It detects violations and reports them to the users through the reporter, and re-evaluates the trustworthy of an application to help user adjusting their security policy.
- Trusted-Degree Calculator (TDC). Traditionally trusted computing take the relationship of trust as an all-or-none problem. But trust, in the real world, is a dynamic, on-changing relationship. One will trust another at some extent in some situation, but don't trust the same entity in some other situation. We use this component to measure the application's trusted-degree in a dynamic way using some machine-learning methods, based on the history of

* Web Services Description Language

interaction between the users and service providers and some other clues.

Three derived utilities are PDP, PEP and access monitor (AM). For each application/user pair, a set of these utilities is prepared to enforce the user's security policy. PDP directs the policy enforcement of PEP and access monitor. PEP will monitor the invoking of an application to the BVM runtime libraries. According to the user's policy, which is decided by PDP, PEP will deny the invoking that violates the expectations of the users. For example, the user can specify the work path to an application. The application should only get access to the specified path. If not, PEP will deny this access request and call RA to report this violation.

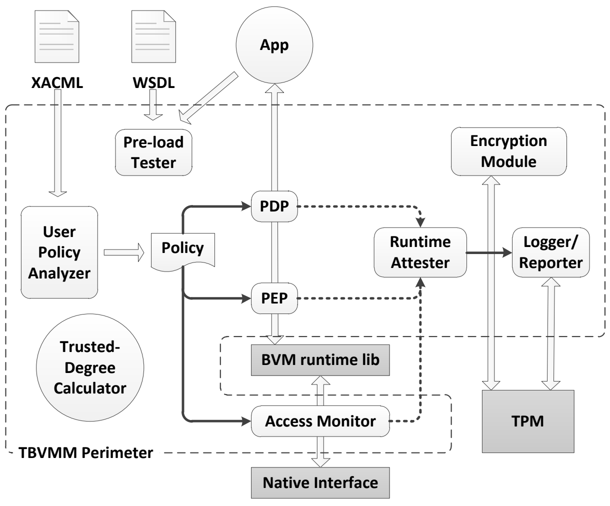

Fig. 2. TBVMM Architecture

The AM does the similar thing with the PEP, except for the AM monitors the native interface invoking. Some applications may call the native interface directly without using the runtime libraries. Still, we cannot totally trust the BVM's runtime libraries and. These libraries may have design pitfalls or even be hacked. Once a tamped BVM API calls a native interface with a distorted parameter, we cannot confine the application's behavior. So we need to pay lots of attention to restrict the native interface.

3.3 *Remote Attestation with TBVMM*

In this section we will specify what attestation can be done with TBVMM, and how attestations are operated.

TBVMM can do nothing to attest the binary code statically. It is designed for attesting the runtime behavior of an application. We use the TBVMM and its associated BVM (including BVM's standard libraries) as the trusted computing base (TCB). TPM should extend its trust chain from OS-level to this TCB.

TBVMM's main objects include:

- Static properties of applications. Users may prefer the applications they used to implement some specific classes or interfaces. These classes of interface may provide the users some important function to fulfill their security or performance needs. So before launching the application, TBVMM can attest the application with vendor's description. After this procedure, the attestation requesters will be sure about whether the application is compatible with their needs.
- Dynamic properties of applications. The application being attested runs under complete control of a TBVMM. Thus, a TBVMM can attest to dynamic properties. This includes the runtime state of the program and properties of the input of the program. Any violations will be reported to the user, user can adjust their security policy according to the application's behavior. Change of an application will cause the degradation of its trusted degree rather than execution halt.
- Specified properties of runtime environments. As mentioned before, TBVMM can monitor the application's working path. Also, it can monitor other environment properties for attestation requesters, e.g. the destination of a network communication, and the peripheral devices being used when the application is running. We can even attest whether a specific protocol (e.g. SSL) has been implemented in the host system. This feature can provide the users a comprehensive vision of the applications.

3.4 *Bayesian-Based Trusted-Degree Estimate*

Trust is a hypothesis about a future behavior. It must be inferred with the current state of the application, the behaviors we now captured, and the policy the application must confine, et. al.. When estimate the future, we must take into account the non-fulfillment of the information, and the uncertainty of knowledge. Therefore we adopt the Bayesian approach as the basic trusted-degree estimating method.

A Bayesian network is an appropriate model which provides a statistic method to calculate the probability of a hypothesis under different conditions. The theoretical background of Bayesian network is the rule:

$$P(h \mid c) = P(c \mid h) \times P(h)/P(c)$$
$$P(h \mid c) \times P(c) = P(c \mid h) \times P(h)$$

For estimating the trust-degree of an application, we build a three-level Bayesian network. As depicted in Fig. 3, the root node of the Bayesian network represents the overall trusted-degree T of an application. The trusted-degree $T \in |0, 1|$, where 0 means the application cannot be trusted at all and 1 means the application can be totally trusted. The children-nodes of the root node represent the basic beliefs of the application. In our model, they are trust of behavior (TOB), trust of state (TOS) and trust of policy confine (TOP). The leaf nodes of the Bayesian network are fine-grinded belief source including most important runtime properties of the application and BVM, the API invoking and each policy described in XACML document.

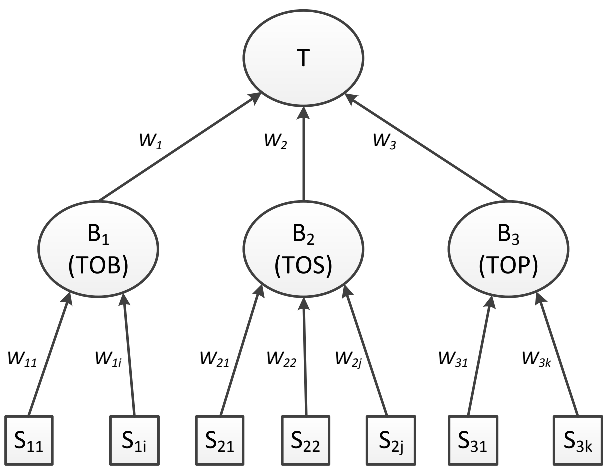

Fig. 3 Bayesian network modeling

We can calculate the trusted-degree of the application using these formulas:

$$T_{app} = \sum_{1 \le l \le 3} W_l \times T_{B_l}$$
$$T_{B_l} = \sum_{1 \le m \le i} W_{li} \times T_{S_{lm}}$$

In the formulas above, W_i is the weight of each basic belief and W_{ij} represents the weight of every belief source.

When estimating the trusted-value of a belief source, we use Kalman filter[18] as the basic mathematic tool. Kalman filter separates the trusted-value estimate into two main phases: Prediction, the state model accounts for the inertia and the erosion of trust based on the propagation of the present trust state, and Correction, the system proposes a revision of the trust state from new observations.

We use $P(x_{k+1} \mid x_k)$ as the Kalman filter's state model, where x_k is the state of a belief source. So we can model the trusted-value estimating process of a belief source with the formulas below:

$$T_{k+1_i} = P(T_{k_i}) \times P(T_{k_i} \mid x_{k+1}) \times P(x_{k+1} \mid x_k)$$
$$P(x_k \mid T_{k_i}) = P(x_k)P(x_k \mid T_{k-1_i})$$

The former one is the prediction process and the latter is the correction process.

With the help of Kalman filter, we can make sure the trusted-degree estimation process itself is trusted and controllable.

3.5 *Implementation and Test Results*

Our implementation is based on the Java Virtual Machine. We used the OpenJDK version 7, running on a quadracore Intel Core-i5 2.66 GHz machine with 4 GB of RAM.

We implemented the six components as several individual modules and expose a set of API for users to invoke these functions. All JVM instances shared these modules, a manage module was added to manage the invoking from different instance. When implement the EM module, we directly call TPM function to encrypt/decrypt the memory data then transport them into JVM's inner memory to speed up the computation process.

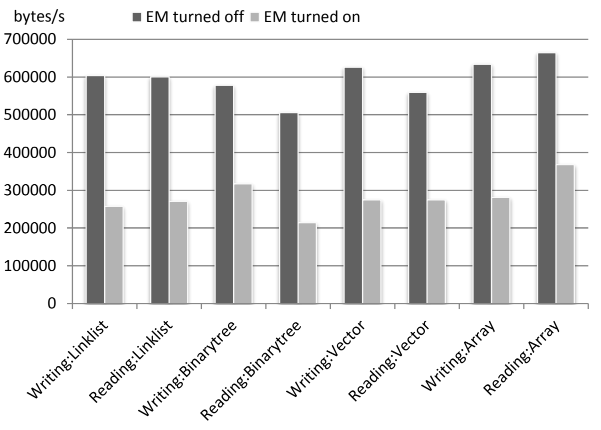

Fig. 4 Performance comparison for memory access. When the memory encryption module is turned on, the read/write performance for every data structure will be almost half as much as that when EM is turned off.

We modified its class loader to add the static property-analysis feature into this virtual machine, this cost about 300 code lines to implement this function.

PDP, PEP, and AM will be embedded into the JVM instance to monitor its dynamic properties and do environment attestation.

We did functional and performance test to determine the achievement of TBVMM. In functional test, we established the security policy and application description and run two file encryption applications. These two applications were implemented with AES and DES algorithms respectively, and they were dedicated to work with different workspace. We chose AES as user's preferred algorithm in the policy file. In the experiment, we saw TBVMM can enforce all the policy restrictions according to our definition. TBVMM can match our design goal in functional aspect.

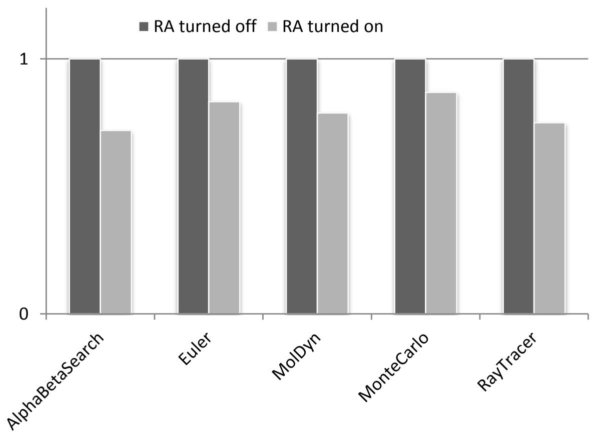

Fig. 5 Performance comparison for user-defined constraints. When the RA and PA modules are turned on, the file system access operations will be examined carefully. So it shows a clear performance loss in this figure..

We used Java Grande benchmarks to test performance impact the TBVMM brings to the JVM. We designed two experiments to do this test. First, we turned off all the core modules except the ME module to test memory read/write cost caused by encrypt/decrypt operations. At the same time, we did not apply any user defined policy to the TBVMM. In this situation, we run the Serial benchmark of Java Grande's Section1 benchmarks. Then we turned off all the core modules and do run the benchmark again. Fig. 4 gives out the tested results. We can see that when EM is turned on, there are significant impacts on the memory-access ability.

Second, we turned on the RA and PA modules and specified the work path of JVM's to the Java Grande's root path. We then did a test with the Section3 benchmarks, which need access the class file contained in Java Grande's root path. The same experiment was done when RA and PA were turned off. Fig. 5 depicts the result of this experiment.

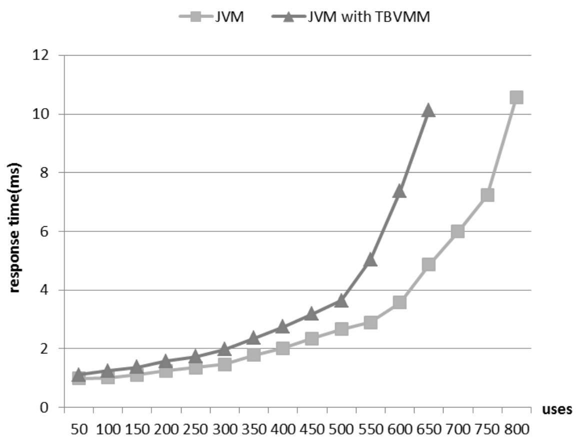

Fig. 6. Concurrency test result. We can see that comparing with the pure JVM, TBVMM causes a significant performance reduction when the user number get to 600.

In cloud environment, multi-users can access the cloud resource in a simultaneous way. So we did some tests to estimate the TBVMM's dynamic concurrency. We run a Tomcat servlet container on original JVM and our JVM respectively and use Loadrunner to do this test. In Fig. 6, we can find that the original JVM has a better response time than our TBVMM armed JVM. When the number of the virtual users rises to 600, our TBVMM causes a significant performance reduction. And the corresponding number of the original JVM is 800.

These modules bring a not to be neglected performance cost to the JVM. It is obvious that they are TBVMM's performance bottlenecks. When we turned on all the protection mechanism, the JVM's performance reduced to about 40 percent of its original performance. Although we consider that the cost is valuable for making the system more trustworthy and secure. We will still pay a lot of attention to the performance tuning of then TVBMM system.

We also do experiments to evaluate the effect of the trusted-degree calculator's. We first turn off the TDC and do some typical attacks to the BVM, and then we turn the TDC on and do the same tests. Fig. 7 shows that when TDC is on TBVMM can detect more potential risk than its counterpart.

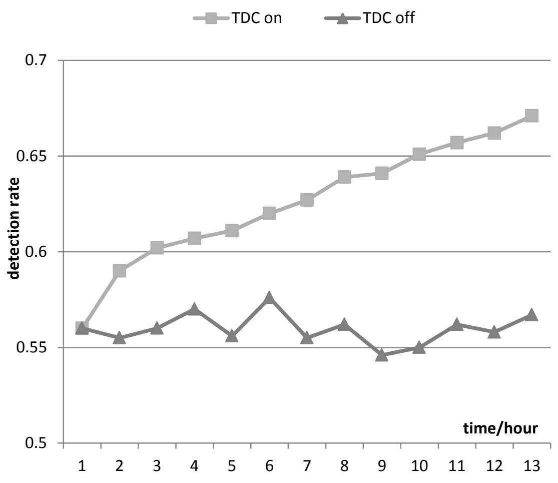

Fig. 7. Risk detection experiments. With the running time increased, the TBVMM with TDC turns on can detect more risk than TBVMM without TDC.

4. Related Work

Some research have been done on trusted computing technology which focus on the improvement of trusted computing technology itself and the application of trusted computing technology to distributed systems and cloud environments.

There is some prior work aims to make the mechanism of remote attestation more fine-grained, dynamic. In Ref. 9, Sailer et al. propose an integrity measurement architecture (IMA) based on TC technology, which extends the remote attestation mechanism to the application layer of the system by maintaining a measurement list in the kernel. Jaeger et al. use CW-Lite model to reduce the IMA architecture[11], so that only the integrity of the trusted subjects and the information flow between them will be monitored, which resolves the problem caused by wrong input data in IMA. This technology is not suitable to attestation for web-based applications which are always run on the top of JVM or CLR. The applications themselves are treated as the input data of the virtual machines'. But can be used to measure the TBVMM and BVM in a dynamic way. Halda et al. propose semantic remote attestation[10], which is very similar to our work. But the semantic remote attestation does not support users to establish their security policy, and cannot fit the various applications in cloud environment. Sadeghi et al. propose the remote attestation based on properties[12]. It provides an alternative to the binary attestation. A Trusted Third Party (TTP) translates the actual system configuration into a set of properties and issues certificates for those properties so that to preserve the system's privacy.

How to combine trusted computing with cloud computing has attracted a great deal of attention recently. Santos et al. propose TCCP[4] a trusted computing framework for IaaS cloud. TCCP define a set of protocol for the IaaS cloud to do trusted initialization and migration of in-cloud virtual machine. Krautheim et al. propose a private virtual infrastructure (PVI) for IaaS cloud[3]. PVI gives users the right of controlling the cloud infrastructure at some extent, and allows organizations to utilize cloud resources with the level of assurance that is required to meet users' confidentiality concerns. In Ref. 5, Krautheim et al. introduce a trusted virtual environment module (TVEM) as the trust root for IaaS cloud, TVEM helps solve the core security challenge of cloud computing by enabling parties to establish trust relationships where an information owner creates and runs a virtual environment on a platform owned by a separate service provider. All these researches are focus on the IaaS cloud model. The need for the trust in PaaS and SaaS cloud are still overlooked. But they provide us a trusted infrastructure to do research on. These technologies can be used to deploy our TBVMM. TBVMM can be used as their trust extension.

A lot of research have been done to solve the problem that how to estimate the trust of an agent or a group of agents. Melaye et al. proposed a Bayesian dynamic trust model for the computational grid[13], but they only give a theoretical model and did not implement their model. Wang et al. propose Cloud-dls, a trust model to make the optimal resource schedule in cloud environment using Bayesian method[14]. Sun et al. proposed an entropy-based trust model for Ad-Hoc network[21]. Theodorakopoulos et al. proposed a semiring-based trust evaluation model and metrics for Ad Hoc Networks[15]. Song et al. proposed a fuzzy trust model for the grid computing system[16]. But these methods are all focus on the reputation and peer-to-peer evaluation in the overall system. They all underestimate the impact to the trust relationship with the state and behavior of the agent itself.

5. Conclusion and Future Work

Conventional ways of remote attestation are based on cryptography. They suffer from many critical shortcomings including static, inexpressive, inflexible et al. Most importantly, they cannot measure program behavior. They can only attest to the presence of a particular binary. Existing dynamic remote attestation technologies can solve some of these problems, but they are not suitable for cloud computing.

Cloud computing bring a tremendous complexity to information security. Attesters can hardly do things efficiently without the help of the cloud environments. Users may lose their control over their critical data and business processes. Cloud should give controls back to the users at some extent. So we propose TBVMM, a novel mechanism for PaaS and SaaS cloud to fill the trust gap between the infrastructure and upper software stacks.

Future work has to be done with other aspects. We will try to add an enhanced isolation mechanism in our TBVMM. With this mechanism, vendors can take advantage of SOA in a more trustable way. This mechanism is also expected to improve the BVM's efficiency when running a big application like CRM and so on.

Acknowledgements

This work was supported in part by the National Natural Science Foundation of China (NSFC) under grant No.60903204.

References

1. M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. H. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, I. Stoica and M. Zaharia, Above the clouds: A Berkeley View of Cloud Computing, *Technical Report UCB/EECS-2009-28*, (EECS Department, University of California, Berkeley, 2009).
2. P. Mell and T. Grance, The NIST Definition of Cloud Computing version 15, *http://csrc.nist.gov/groups/SNS/ cloud-computing*, (National Institute of Standards and Technology. Gaithersburg, MD, 2009).
3. F. J. Krautheim, Private Virtual Infrastructure for Cloud Computing. *In Proc. 2009 Workshop on Hot Topics in Cloud Computing.* (USENIX Association, Berkley, 2009).
4. N. Santos, K. P. Gummadi and R. Rodrigues, Towards Trusted Cloud Computing, *In Proc. 2009 Workshop on Hot Topics in Cloud Computing.* (USENIX Association, Berkley, 2009).
5. F. J. Krautheim, D. S. Phatak and A. T. Sherman, Introducing the Trusted Virtual Environment Module: A New Mechanism for Rooting Trust in Cloud Computing. *In Proc. TRUST 2010. LNCS, 6101*, (Springer, Heidelberg, 2010), pp. 211–227.
6. Trusted Computing Group. TPM Specification Version 1.2. *http://www.trusted-computinggroup.org/resources/ tpm_main_specification*. (Trusted Computing Group, 2007).
7. M. Martin, The Ten-Page Introduction to Trusted Computing. *Research Report CS-RR-08-11*, (Computing Laboratory, Oxford University, Oxford, 2008).
8. K. J. Biba, Integrity considerations for secure computer systems. *Technical Report MTR-3153*, (Mitre Corporation, Bedford, 1975).
9. R. Sailer, X. Zhang, T. Jaeger and L. Doorn, Design and Implementation of a TCG-based Integrity Measurement Architecture. *In Proc. 13th USENIX Security Symposium*, (USENIX Association, Berkley, 2004), pp. 223–238.
10. V. Haldar, D. Chandra and M. Franz, Semantic Remote Attestation: a Virtual Machine Directed Approach to Trusted Computing. *In Proc. 3rd Conference on Virtual Machine Research and Technology Symposium*, (USENIX Association, Berkeley, 2004).
11. T. Jaeger, R. Sailer and U. Shankar, Prima: policy-reduced integrity measurement architecture. *In Proc. 11th ACM symposium on Access control models and technologies*, (ACM Press, New York, 2006), pp. 19–28.
12. A. R. Sadeghi, and C. Stuble, Property-based attestation for computing platforms: caring about properties, not mechanisms. *In Proc. 2004 workshop on New security paradigms*, (ACM Press, New York, 2004), pp. 67–77.
13. D. Melaye and Y. Demazeau, Bayesian Dynamic Trust Model, *Multi-Agent Systems and Applications*, (Springer, Heidelberg, 2005), pp. 480-489.
14. W. Wang, G. Zeng, D. Tang and J. Yao, Cloud-DLS: Dynamic trusted scheduling for Cloud computing, *Expert Syst. Appl*. 39(3) (2012) 2321-2329.
15. Y. Sun, W. Yu, Z. Han, K. J. R. Liu, Trust Modeling and Evaluation in Ad Hoc Networks, *In Proc. IEEE Global Telecommunications Conference*, (IEEE Press, 2005), pp. 1862-1867.
16. G. Theodorakopoulos and J. S. Baras, On trust models and trust evaluation metrics for ad-hoc networks, *IEEE Journal on Selected Areas in Communications, 24(2)* (IEEE Press, 2006), pp. 318–328.
17. S. Song, K. Hwang and M. Macwan, Fuzzy Trust Integration for Security Enforcement in Grid Computing, *Network and Parallel Computing*, (Springer Berlin 2004), pp. 9-21.
18. G. Welch and G. Bishop, An Introduction to the Kalman Filter, *Technical Report 95-041*, (Department of Computer Science, University of North Carolina at Chapel Hill, 1995).