T-YUN: Trustworthiness Verification and Audit on the Cloud Providers

Chuanyi LIU†, ‡, Member, Jie LIN∥, ‡, and Binxing FANG†, ‡ ‡, Nonmembers

SUMMARY Cloud computing is broadly recognized as as the prevalent trend in IT. However, in cloud computing mode, customers lose the direct control of their data and applications hosted by the cloud providers, which leads to the trustworthiness issue of the cloud providers, hindering the widespread use of cloud computing. This paper proposes a trustworthiness verification and audit mechanism on cloud providers called T-YUN. It introduces a trusted third party to cyclically attest the remote clouds, which are instrumented with the trusted chain covering the whole architecture stack. According to the main operations of the clouds, remote verification protocols are also proposed in T-YUN, with a dedicated key management scheme. This paper also implements a proof-of-concept emulator to validate the effectiveness and performance overhead of T-YUN. The experimental results show that T-YUN is effective and the extra overhead incurred by it is acceptable.

key words: cloud computing, trustworthiness, audit, attestation, trusted computing

1. Introduction

Cloud Computing is a commercial network-based computing service model, which provides resources with the need-on-demand manner. Typical characteristics of cloud computing includes large-scale, virtualization, versatility, on-demand services and high scalability [1]. Cloud computing represents the trend of IT field quickly develop to the features of intensive, large-scale and specialized, enabling IT services more cost savings and more efficient use of computing resources [2]. So cloud computing is generally regarded as the next important IT industry growth point following the booming of Internet economic, and it will have a huge market and growth prospects.

A typical cloud computing environment consists of three independent roles as: the cloud provider, the service provider and customers. The cloud provider manages and virtualizes physical resources, and provides service to the service providers or directly to customers usually in the form of virtual machines (VM). This kind of service is also Infrastructure as a Service (IaaS), based on which the service providers can build a variety of specific services to customers [3].

The ubiquitous adoption of cloud services depends largely on the trustworthiness of the cloud providers [4], [5], because customers lose direct control of their data and applications hosted in cloud environments. And the data and applications may be subject to two kinds of threats: insider attacks such as malicious staff or administrator internal of the cloud providers [6] or attacks from other cloud users sharing the same cloud computing infrastructure and resources [7].

Specifically, a trusted cloud provider must ensure the integrity and trustworthiness of the whole architectural stack in corresponding physical servers that Virtual Machines (VM) are assigned or migrated to, such that VMs are always in a trusted environment. Moreover, as the cloud customers use VMs, the execution environment within the VM should also be guaranteed.

There are mainly two approaches at present to solve the above problem summarized as follows:

The first approach only protects the user data, with assumption of untrusted execution environment [9]. Recently homomorphic encryption has received special interest [8], which can map algebraic operation of plaintext to particular operation of ciphertext, so as to ensure the cloud provider can still execute compute or manage task on the ciphertext without decrypting the plaintext.

The other approach directly focuses on the trustworthiness of cloud provider, and this approach receives more attentions both from industry and academic. Specifically, as VMs are the basic services provided by cloud providers, a trusted cloud provider should ensure the integrity and trustworthiness of all the entities involved in the process that the VMs assigned to and run on a node or migrate to other node when necessary, which will guarantee that the VM is always running in a trusted environment. Based on whether the introduction of a Trusted Third Party (TTP), the existing cloud platform trusted model is divided into two categories, namely: 1) build a trusted cloud platform, to prove its trustworthiness directly for users; 2) the introduction of a TTP to audit and verify the trustworthiness of cloud providers.

There are several challenges to assure the trustworthiness of cloud providers, which can be summarized as follows:

- Since the management responsibilities are divided between customer and provider, neither of them is in a good position to address these problems. Even detecting the presence of a problem can be surprisingly difficult: on the one hand, the provider does not know what
to look for, since he does not know what the computation is supposed to do; on the other hand, the customer can only access the cloud machines remotely, so he has only very limited information.

- When a problem is detected, the customer and the provider face the potentially difficult task of deciding which of them is responsible for it — it is quite natural for the provider to initially suspect a problem with the customer’s software, and vice versa. And if such a dispute cannot be resolved amicably, it is nearly impossible for either of them to convince a third party (such as an arbitrator or a judge) that the other is responsible.

- Different cloud providers’ platform configuration environment is different, it is difficult to provide a common measurement to verify the trustworthiness of different cloud platforms and do not too many restrictions on the cloud platform.

Based on trusted computing technologies [16], this paper proposes T-YUN, which introduces a TTP to attest and audit the trustworthiness of the cloud providers periodically. Specifically, T-YUN handles the whole architecture stack, extending current trusted chain into the virtual machines, which are the actual execution environments for the cloud users. Different from most of state-of-art works, T-YUN adopts a cyclic verification protocol to guarantee the trustworthiness during run time. A dedicated key management scheme is also designed in T-YUN, which is represented in the remote verification & audit protocols.

This paper is organized as follows: the first part discusses the latest related works; the second part summarizes the threat model; the third part introduces the trustworthiness audit and verification mechanisms; the fourth part describes the audit and verification protocols, which ensures that the VM always running in a trusted environment throughout its life cycle; the fifth part illustrates the effectiveness and analyze the cost of the mechanism through the experimental data; Section 6 summarizes the content of this paper.

2. Related Work

There are two kinds of verification model for trusted cloud providers: (1) build a trusted cloud platform, and prove its trustworthiness directly to users; (2) the introduction of a TTP to audit and verification cloud provider trustworthiness. These two methods all need to use trusted computing technology to collect the credible evidence, based on the TCB (Trusted Computing Base) to build a trust chain of cloud platform. The followings separately analysis related work from the points of above two kinds of model:

The work of building a trust cloud platform as follows:

CERTICLOUD [10] proposed two protocols to verify the trustworthiness of cloud: TCRR and Verify MyVM. When the first one asserts the integrity of a remote resource and permits to exchange a private symmetric key, the second authorizes the user to detect trustfully and on demand any tampering attempt on its running VM.

Some researchers ensure the trustworthiness of cloud by verifying the trustworthiness of node hosting the VM. Joshua Schifman [11] establish CV (Cloud Verifier) services to provide users with the integrity evidence to verify the integrity of the cloud platform and access control execution capabilities, ensuring the integrity of the user’s application host on VM of IaaS cloud.

IBM puts forward the TVDs (trusted virtual domains) technology based on trusted computing technology [12], and combined with MAC (Mandatory Access Control) strategy to provide strong isolation measure for the different cloud user tasks. This method ensures cloud users’ data will not disclose to each other and viruses or malicious code cannot spread mutually. At the same time, TVDs can also verify the integrity of user’s software run in the cloud platform.

The works that introduce the TTP are as follows:

Nuno Santos [13] first proposed the use of trusted computing technology to build a trusted cloud system, and designed a trusted cloud computing platform TCCP, which consists of two components: TVMM (Trusted VM Management and TC (Trusted Coordinator). TCCP provides a closed box execution environment that guarantees confidential execution of guest VMs, moreover, TCCP introduces TTP for users to attest to the IaaS provider and determine whether or not the service is secure before they launch their VMs.

RepCloud [14] treats the cloud as a P2P network, using of reputation system evaluation mechanisms to deal with the cloud dispersed verification process, and gave a trustworthiness degree range in a certain value.

Imran Khan [15] designed and deployed a Trusted Eucalyptus Cloud by introducing a TTP named of TIV (Trusted Integrity Verifier), make sure that the users’ VM can only start when it meets the integrity verification of nodes.

3. Threat Model

In order to properly design and evaluate a trusted cloud computing system, the threat model must be clearly laid out. In this section the present and potential adversaries, thread model assumptions in our architecture, and the attacks are identified.

Cloud providers offer VM to users as the core business, and also provide virtual disk, virtual network and online storage. The threat model of trusted cloud providers can be classified into the followings: (1) all of the services are dependent on the physical servers, and components of these servers, such as hardware, BIOS, Bootloader, OS, VM manager, etc., may be attacked or maliciously modified in the start process or operation; (2) the cloud system may be attacked by unauthorized malicious users, or there exists malicious administrator has the privilege, who will achieve cloud system or user data through the way of installing trojans or the back door program (3) The VM image provided by cloud may be suffered from middleman attack in the transmission process, the VM instance operating may also
be replaced or removed to the malicious servers; (4) cloud providers may intentionally or unintentionally distribute insufficient resources to users, against the service level agreement (SLA).

However, system administrators often have physical access to servers, which can take some advanced means of attack, such as cold boot attack or even tampered with hardware, but for the general cloud provider, there will be no such administrator have all these privileges, meanwhile, cloud provider will take strict access control strategy to protect the integrity of the physical hardware. Therefore, we assume that cloud providers can prevent attacks need physical access to the server.

4. Trustworthiness Verification and Audit

4.1 System Model

As shown in Fig. 1, T-YUN introduces TTP to collect the trusted evidence of the cloud servers and take charge of the audit and verification of cloud providers. There are two kinds of servers in a typical cloud provider, as: controllers and nodes. Controllers are responsible for cloud management, such as metadata management, user management, security management and interconnection for the cloud components, while nodes are the hosts of virtual machines, which are running on top of the hypervisors.

The trustworthiness verification system of T-YUN involves three roles: TTP, Configuration List Collection (CLC) module and Remote Attestation Service (RAS) module within the node. TTP verifies the trustworthiness of physical servers in cloud. Only after pass through the verification of TTP, the node would provide a trusted host operating environment for VM, and ensure to detect the destruction of integrity of VM timely. CLC module running in the OS kernel mode, which is the first loaded after the system boot, so that it can collect the digest value of other programs. RAS module interacts with TTP to receive the attestation request and return related information of node.

4.2 Node Architecture

In order to use trusted computing technology to audit and verify the trustworthiness of cloud provider, cloud provider’s nodes must provide the necessary support both in hardware and software.

Typically, the hardware of node needs to be equipped with TCG Trusted Platform Module (TPM) chip as a trusted root. To software, BIOS and Bootloader need to add trust measure function that the SRTM specification demands, the OS kernel needs to add configuration collected module (CLC), and deploying trusted remote attestation service module (RAS) at the application layer, which will rely on TCG software stack (TSS) to call TPM function. These configurations of hardware and software are only functional requirements, without too many restrict demands of its manufacturer, version, etc.

If cloud provider upgrades its nodes configuration according to business need, it simply registers the new configuration information to TTP and pass through its verification. To some extent, this mechanism solves the problem of existing trustworthiness verification and audit models make too restricted configuration demands on cloud providers.

4.3 The Verification Mechanism

The trustworthiness verification consists of two processes: the evidence collection process and remote attestation process.

In the evidence collection process, the CLC module of the node calculates the hash digest of the programs or scripts to be loaded into memory for execution, and saves the digest values to the configuration list (CL), while extending
the hash digest to TPM PCR (Platform Configuration Registers) in order to ensure the integrity of the CL.

In remote attestation process, TTP receives the related verification information of node, including: CL, PCR, and AIK certificate that will provide AIK public key to prove the identity of TPM of node. TTP uses Finite State Machine (FSM) to predefine the trusted behavior sequence, if the system behavior not beyond the scope of FSM, then it is untrusted. TTP will not only check the consistency of CL and PCR, but also examine the system behavior recorded in CL with FSM.

The detailed verification procedure is shown in Fig. 3, where TTP firstly generates a nonce, and sends it to RAS module of the node. Upon receiving the nonce and loading it into TPM, the RAS module uses TPM to quote the current PCR value and nonce; then it return the TPM signed result, AIK certificate and CL to TTP. When TTP receives the evidence information, it first use AIK certificate to verify the signature of the TPM, which also prove the identity of TPM. Then TTP checks the nonce which have been received with the one sent before, ensuring the received verification information is in the same session. TTP next uses CL to analog extend operation to calculate the PCR value, the calculate method is: $PCR' = SHA1(PCR'||MValuei)$, $MValuei$ is one of the measurement in CL. $PCR'$ is the final value needs to compare with PCR value. If the calculate PCR value consistent with the returned PCR value, it indicates that the CL has not been modified; Finally, TTP retrieves all the hash digest of CL in the trustworthiness program database to find out whether there is any malicious program exists or unknown program, if so, the verification result is that the server is untrusted; else, the node is trusted.

In this paper, the trustworthiness verification and audit system has the following features:

1. Use FSM to predefine the trusted system behavior, which can detect the malicious system behavior sequence.
2. TTP takes the periodic verification way to effectively solve the problem of dynamic trustworthiness verification, which is one of the major challenges of trusted cloud. The periodic verification time can be adjusted as needed.

TTP maintain a unified dynamic trustworthiness software warehouse, in which records the integrity of the information of all kinds of trusted software. If some software upgrade, the warehouse should be synchronously updated, solving the problem of the clouds dynamically upgrade in the trustworthiness verification.

5. Trustworthiness Verification Protocols

Cloud provider provides users VM as the core business, so this paper designs the trustworthiness verification protocols related to the VM lifecycle management, which including four trust attestation protocols: node registration protocol, node boot protocol, VM boot protocol, VM migration protocol. These four protocols can ensure the trustworthiness of user’s VM in all its lifecycle.

Compared with related works, the verification protocols in T-YUN have the following characteristics:

1. The trust attestation mechanism considers all the lifecycle of VM to achieve a rigorous attestation result, rather than only the attestation protocol when the VM running, which will lost its trustworthiness when in other status.
2. Design a key management system of TTP combined with these protocols. Specifically, rationally manage the usage time and the usage frequency of the TPM keys, for example, EK is only used for node boot protocol to generate AIK; As node identity, the AIK only used for the signature operation of the TPM internal data (random number, the TPM state information and other keys), the session keys are used to encrypt and decrypt the sensitive data in the communication process.

5.1 Node Registration Protocol

There is an assumption of this protocol, that is, the node registered first time is trusted, such as the node just finish installing the operation system. The process is launched and triggered by TTP, which establishes a SSH connection with the node registered, which forms a secure channel for communication to ensure that data will not compromised, and the compression of data will speed up the transmission process. Then, TTP sends message to request the trust configuration information of node, which is not compromised and can reflect the trust status of node. After receiving the request, the node collects local configuration information, including: UUID of node, public key as well as database information, the database store hash value and name of programs running on the node. Finally node returns configuration information back to TTP. The protocol is depicted in Fig. 4.

5.2 Node Boot Protocol

We use the pre-defined physical server behavior model to determine whether the boot sequence of the physical server

![Fig. 3 Trusted remote attestation mechanism.](image-url)
is trustworthy or not. The trustworthiness of programs, opened files, and loaded kernel modules on the physical server can be checked by comparing their hash digests in the configuration file to the feature database maintained by the TTP. The TTP can further inspect the vulnerabilities on the physical server according to corresponding requirements. Based on the life cycle of VMs, remote verification is triggered by events, such as VM migration or VM creation, and timer. The evidence collector accesses TPM to get the current PCR values and configuration file, and the communicator transfers the above information to the TTP, which comparing them with its local database to guarantee the VM runs in a trusted hosting environment.

After booting up the operating system of the node, it sent a request $Nonce_N$, which is used to identify a typical session from the node side, to begin the trust attestation. Upon TTP receiving the request, the TTP encrypts $Nonce_N$, and generate another nonce $Nonce_TTP$, which is used to identify a typical session from the TTP side, append to encrypted $Nonce_N$, and then sent back to the node. The node verifies $Nonce_N$ to ensure the security of the current session, use the private AIK key to quote CL, and finally sent to TTP.

TTP saves the trust CL of the node. In the later remote validation phase, TTP will compare trust CL with the realtime CL of node to verify the trustworthiness of the nodes.

### 5.3 VM Launch Protocol

The core service of IaaS cloud is to provide VMs to users, so it is very critical process that opening the VM. This paper makes improvement to ensure the trustworthiness of the VM startup process. The VM must start up on a trust node of cloud, which have register and verified by TTP.

The roles involved in the process of VM launch include: user, control node, register node, TTP. The process as following:

1) User fills out CL of VM, such as the core number of the CPU, memory size, mirror type, the number of VMs.
2) User generates a symmetric session key $K_{VM}$, which is used to encrypted hash value of VM image; then use TTP public key to encrypt this key, ensuring that only TTP can read it.
3) User calculates the hash value of the selected mirror, then use $K_{VM}$ to encrypt the hash value.
4) Send the CL of VM, the encrypted session key $K_{VM}$, the encrypted VM images and its hash value to the control node of T-YUN through a secure channel.
5) Control node will select an appropriate trusted node to start the VM according to the user request information of VM. The appropriate node must meet the following requirements: Have been registered on TTP and verified by TTP, the remaining number of CPU cores is greater than the request; the remaining memory capacity is greater than the requested VM memory size. After selecting node, control node will resend all the information the user request to the register node.
6) Registered node receives the request, first need to achieve $K_{VM}$ to decrypt the VM images, so register node send the encrypt $K_{VM}$ to TTP.
7) TTP receives the request, first need to verify the trustworthiness of the registered node send this request.
8) If the verification result is trust, then it decrypt the $K_{VM}$ and send it back to register node.
9) Registered node receives the $K_{VM}$, using this key to decrypt the VM image and its hash value.
10) Calculate the hash value of VM image, compare it with the hash value that decrypted, if they are equal, the VM image is integrity.
11) Then the register node run the VM that user request through VM Manager (KVM or Xen).
12) After the VM turned on, the registration node collects the actual configuration of the VM, and returned it to the control node.
13) Control node forwards the VM configuration information to the user.
14) User receives the CL of VM, they will know the trust-
worthiness of the entire VM’s boot process.

5.4 VM Migration Protocol

In order to guarantee the reliability and availability of virtual server status and user data, the VM status should be migrated across different hosts in case of various host failures. Live migration provides an approach to achieve the above goal transparently. To make this operation trusted, both the source node $N_s$ and the destination node $N_d$ should be trusted, and the VM status should remain confidential and unmodified while it is transferred over the network.

The procedure of VM migration as followings:

1) Control node selects the appropriate node $N_d$ according to the migration demand. The standard of the appropriate node is as same as the node in the process of VM lunch. Then Control node sends the information of $N_d$ to $N_s$.

2) $N_s$ receives information from $N_d$, it sends the request that need to achieve public key $TKP_{N_d}$ of $N_d$ to TTP. $TKP_{N_d}$ is used to encrypt a session key $K_S$, which is used to encrypt the migration VM mirror and its hash value. The purpose is to ensure that only the $N_d$ can achieve the information of the migration VM mirror.

3) TTP receives the request of $N_s$, it verifies the trustworthiness of both the $N_s$ and $N_d$.

4) If $N_s$ and $N_d$ are all trusted, then TTP sends $TKP_{N_d}$ to $N_s$.

5) $N_s$ generates $K_S$, then uses $TKP_{N_d}$ to encrypt $K_S$.

6) $N_s$ uses $K_S$ to encrypt the migration VM mirror and its hash value.

7) $N_s$ sends the encrypted $K_S$ and the encrypted information of VM to $N_d$.

8) $N_d$ uses its private key to decrypt the $K_S$.

9) $N_d$ use $K_S$ to decrypt the VM mirror and the hash value.

10) $N_d$ calculates the hash value of VM image mirror, compare it with the hash value that decrypted, if they are equal, the migration VM image is integrity.

11) After the verification, $N_d$ resumes the VM.

6. Evaluation

The goals of evaluation include the following two aspects: firstly, experimentally evaluate the effectiveness of T-YUN trustworthiness verification mechanism; and the extra performance cost introduced by T-YUN. To figure out the performance costs, we use three types of VMs in our experiments: trusted VM, semi-trusted VM, and non-trusted VM.

The trusted VM is implemented with a virtual TPM and IMA, which is used to calculate the integrity of the software and hardware configurations. The semi-trusted VM is implemented with only IMA without a virtual TPM. The non-trusted VM is a common VM without virtual TPM and IMA. The macro-benchmarks such as PostMark and TPCC-UVa are used to test the processing capabilities of file and database transactions, and micro-benchmarks such as ByteMark and IOzone are used to test the performance of computing and I/O operations.

With the purpose of just want to evaluate the effectiveness and performance of T-YUN, and not really use it, it is not necessary to build T-YUN consists of nodes equips with real TPM, which will be high cost and troublesome to change the hardware configuration as needed. So we designed and implemented an emulator, using VM with vTPM to instead really nodes of T-YUN to build the simulate T-YUN, which will be flexibility to configure and specify the scale.

6.1 Prototype Implementation and Experimental Setup

The architecture of VM emulator is shown as Fig. 8: In host OS, QEMU acts as the VMM (VM Monitor), running VM with the KVM accelerate module. The vTPM (virtual TPM) module of QEMU is the role of TPM hardware form the point view of VM. SeaBIOS is an open source implementa-
tion of X86 BIOS, plays the role of BIOS of VM and fulfills the function of the initialization of TPM, SRTM (Static Root of Trust Measurement) and the extend of TCG BIOS.

GRUB-IMA is the enhancement of Linux boot loader adding the TCG measurement capability. It measures the grub during the process of loading Grub.

IMA acts as CLC module of T-YUN nodes, works by hooking into the mmap() operation. Whenever a file is mapped in an executable mode, the IMA hook will first perform and save the SHA1 digest in CL file.

The Trousers realizes TSS (TCG Software Stack), providing an API to operating system and applications so that they can use the functionality provided by a TPM.

OpenPTS is the open source implement of TCG PTS (Platform Trust Services) standard. The collector of openPTS acts as RAS module of T-YUN nodes, while the verifier of openPTS running in TTP to finish trustworthiness attestation task.

6.2 Experimental Results and Analysis

(1) Effectiveness Evaluation

In this section, we use two typical malicious programs to test the effectiveness of the T-YUN, as: Logtamper v1.0 and Mood-nt 2.3. The former is an application-level program, which can maliciously modify the log files of Linux system. The latter is a kernel-level rootkit program. We implanted these two malicious programs into one running node of T-YUN to test whether T-YUN can detect them.

As shown in Fig. 9, “pid” in the first column is the ID of the process; “name=/wtmpclean” or “name=/mood-nt” shows the name of malicious programs which are logged in the configuration list. “user=root” in the third column shows the process’ or the kernel module’s privilege. The value in the fourth column is the integrity digest of the programs. And the attestation result is listed in the last column, with “result=UNWANTED” showing that the program is untrusted. The above results illustrate that T-YUN can effectively detect both application-level and kernel-level malicious programs.

To further evaluate the effectiveness of T-YUN about its unique features. We use the most common attack on loaded programs: buffer overflow attacks, as examples to illustrate the unique effectiveness of T-YUN. Before a buffer-overflow attack, the stack structure of our example code is shown as Fig. 10 a). The main function has only a local variable named ‘buffer’, which is a char array that has 96 bytes. The stack and the array grow in a reverse order. The beginning address of the array is at the bottom of the array. As the string functions and standard input and output functions in libc do not check the length of the array, a simple buffer overflow can overwrite the return address by writing data longer than 96 bytes.

In such an attack, we write data that has 128 bytes to the buffer declared as 96 bytes. The first 96 bytes contain a malicious shell that we want to execute, and the remaining 32 bytes contain the address of the buffer that we use to overwrite the original return address of the main function. When the main function returns, the process will execute
LIU et al.: T-YUN: TRUSTWORTHINESS VERIFICATION AND AUDIT ON THE CLOUD PROVIDERS

2351

Fig. 13 Quantitative analysis of the performance cost introduced by T-YUN. (a) Ratio of execution time for typical Linux Commands; (b) Computation Performance Comparison using BYTEmark benchmark; (c) Small I/O performance comparison using PostMark benchmark; (d) I/O Transaction Throughput comparison using TPCC benchmark.

By running the code like ours, an attacker can change the return address of victims’ code and running malicious code as shown in Fig. 11. To show the result, we printed the return address before the buffer overflow to inject malicious code.

After injecting the shell, we run our detection module to get the return address from the stack of main function, as shown Fig. 12. By comparing with the original address, we know a buffer overflow has occurred.

(2) Performance Cost Evaluation

In this section, we use micro-benchmarks and macro-benchmarks to evaluate the performance cost of T-YUN. The quantitative results are measured from the users’ perspective, i.e. the overhead measured within users’ VMs. We denote the VMs under T-YUN mechanism as TVM (Trusted VM), while the other VMs without T-YUN as ordinary VM. Figure 13 shows the performance contrast between TVM and VM.

The evaluations of micro-benchmark are shown as the two charts on top of Fig. 13, reflecting the performance difference between TVM and VM in command execution and computing ability. The two charts on bottom of Fig. 11 belong to macro-benchmark, showing the file processing ability and database transaction ability difference of TVM compared with VM.

We selected 10 most commonly used Linux commands to test the ratio of execution time in TVM and VM. It can be seen from the chart that, the average time of command execution of TVM is longer than the VM by 20%. However, the time is in milliseconds level, so the effects can almost be neglected.
The performance overhead incurred by the verification protocols in T-YUN is proposed. It introduces this paper, T-YUN: a trustworthiness verification and audit mechanism on the cloud providers. In Whether the cloud computing can be widely used is largely determined by the trustworthiness of cloud providers. In this paper, T-YUN: a trustworthiness verification and audit mechanism on the cloud providers is proposed. It introduces a TTP to cyclically verify and audit the trustworthiness of remote clouds.

The characteristics of T-YUN include:

- A holistic trusted chain, from physical infrastructure to user virtual machines, is constructed for the cloud. And the whole architecture stack is attested and audited by the remote TTP.
- A dedicated key management scheme is also designed in T-YUN, which is represented in the remote verification protocols.
- A cyclic attestation scheme is proposed in T-YUN to guarantee the trustworthiness during run time.

Experimental results show that the performance overhead introduced by T-YUN is acceptable.

Acknowledgements

This work is supported by the China NSFC Program under Grant No. 61202081.

References

[1] M. Armbrust, A. Fox, R. Griffith, A.D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, “Above the clouds: a Berkeley view of cloud computing,” USA: University of California at Berkley, Technical Report: UCB/EECS-2009-28, 2009.

[2] M. Armbrust, A. Fox, R. Griffith, et al., “A view of cloud computing,” Commun. ACM, vol.53, no.4, pp.50–58, 2010.

[3] V. Jinesh, “Migrating your existing applications to the AWS cloud,” http://www.Amazon.com, Oct. 2010.

[4] Y. Chen, V. Paxson, and R. Katz, “What’s new about cloud computing security?,” Berkeley USA: University of California at Berkley, Technical Report: UCB/EECS-2010-5, 2010.

[5] R.K.L. Ko, P. Jagadpramana, and M. Mowbray, et al., “TrustCloud: A framework for accountability and trust in cloud computing,” 2nd IEEE World Congress on Services, pp.584–588, Washington, DC, 2011.

[6] W. Jansen and T. Grance, “Guidelines on security and privacy in public cloud computing,” National Institute of Standards and Technology, Gaithersburg, MD, USA, 2011.

[7] S. Marco, “BlackHat presentation demo vids: Amazon, part 4 of 5, AMIBomb[EB/OL],” http://www.sensepost.com/blog/3797.html, Aug. 8, 2009.

[8] G. Craig, “Fully homomorphic encryption using ideal lattices,” 41st ACM Symposium on Theory of Computing (STOC), pp.169–178, 2009.

[9] 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,” Proc. 3rd international congress on Trust and trustworthy computing, pp.211–227, Berlin, Germany, 2010.

[10] B. Bertholon, S. Varrette, and P. Bouvry, “Certicloud: A novel TPM-based approach to ensure cloud laas security,” Proc. 4th International Conference on Cloud Computing, p.121, Washington, USA, 2011.

[11] J. Schiffman and T. Moyer, “Vijayakumar hayawardh, jaeger trent, McDaniel patrick. Seeding clouds with trust anchors,” Proc. 2010 ACM Workshop on Cloud Computing Security Workshop, Chicago, Illinois, USA, pp.43–46 2010.

[12] S. Berger, R. Cáceres, D. Pendarakis, R. Sailer, E. Valdez, R. Perez, W. Schildhauer, and D. Srinivasan, “TVDC: managing security in the trusted virtual datacenter,” ACM SIGOPS Operating Systems Review, vol.42, no.1, pp.40–47, 2008.
[13] R. Neisse, D. Holling, and A. Pretschner, “Implementing trust in cloud infrastructures,” Proc. 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing. Los Alamitos, pp. 524–533, California, USA, 2011.

[14] R. Anbang and M. Andrew, “RepCloud: Achieving fine-grained cloud TCB attestation with reputation systems,” Proc. 6th ACM workshop on Scalable trusted computing, pp. 3–14, New York, USA, 2011.

[15] I. Khan, H. Rehman, and Z. Anwar, “Design and deployment of a trusted eucalyptus Cloud,” 2011 IEEE International Conference on Cloud Computing (CLOUD), pp. 380–387, Washington, USA, 2011.

[16] Trusted Computing, Accessed from http://en.wikipedia.org/wiki/Trusted_Computing, Feb. 2013.

Chuanyi Liu is now an assistant professor of computer science & engineering at Beijing University of Posts and Telecommunications. He received his Ph.D. (2009) in computer science and technology from Tsinghua University. He has broad research interests in computer systems, including Computer Architecture, Cloud Computing & Cloud Security, File and Storage Systems, Information Security and Data Protection. He spent one year as a visiting scholar at the Digital Technology Center of the University of Minnesota.

Jie Lin is a doctoral candidate at Beijing University of Posts and Telecommunications. He received the master’s degree (2011) in computer application technology from Yunnan University of Finance and Economics. His current research interests include computer network, information security.

Binxing Fang is the president of Beijing University of Posts and Telecommunications. He is also the Member of Chinese Academy of Engineering. He received the doctor’s degree (1989) in computer science and technology from Harbin Institute of Technology. His current research interests include computer network, information security.