NeuCloud: Enabling Privacy-preserving Monitoring in Cloud Computing

Yulong Zhang, Min Li, Benjamin Wilder and Meng Yu
Department of Computer Science, Virginia Commonwealth University
401 West Main Street, Richmond, VA 23284
{zhangy44,lim4,wilderbw,myu}@vcu.edu

Kun Bai
IBM T.J. Watson Research Center
19 Skyline Dr., Hawthorne, NY 10532
kunbai@us.ibm.com

Peng Liu
College of Information Sciences and Technology
Pennsylvania State University
IST Building, State College, PA 16802
pliu@ist.psu.edu

ABSTRACT

Cloud computing has changed how services are provided and supported through the computing infrastructure. However, the lack of trust from both sides - the client and the cloud provider, is still a major obstacle to prevent many users from using the cloud, especially for users who are very concerned about the privacy of their computing and data in the cloud. There has been little research on architectures designed to monitor client VMs while preserving their privacy in cloud computing.

Based on the argument that service provider needs only the monitoring and management functionalities instead of a direct ownership of them, we propose a new architecture, called NeuCloud, for trusted neutral cloud computing. The new architecture provides an unprivileged service domain to the service provider and deprive the direct monitoring and management privilege of him. Instead, a neutral domain, as part of a mutual agreement between the client and the service provider, supports security monitoring and cloud management with minimum attack surface. As a result, privacy-preserving monitoring can be enabled in NeuCloud. In this paper, we introduce the NeuCloud design and describe how to implement NeuCloud in both Type-I and Type-II virtualization. Our experiments show that the overhead of NeuCloud is negligible.

Keywords

Trust, Monitoring, Privacy, Cloud Computing

1. INTRODUCTION

Cloud computing [5] is becoming a major trend in computing services with its inspiring features of elastic "data anywhere" and "computing anywhere". Meanwhile, because services are carried out in a form where customers do not directly manage their private data, cloud computing has also been the subject of much public scrutiny concerning issues of privacy protection. According to a survey result, concern about the possibility of privacy leakage has become the most critical reason that hinders a broad adoption of Cloud Computing [2]. Additionally, even if the clients themselves can trust the cloud provider, some privacy related laws restrict a business' freedom to outsource their sensitive computing to cloud providers [11]. Therefore, many new security mechanisms have been developed to protect users' privacy. Some techniques protect data privacy through cryptography, such as the techniques in [7]. However, data processing over encrypted data is limited. Other techniques like [16, 36, 29, 34, 10, 19, 25] protect VMs from being inspected by the service provider. However, service providers are averse to utilizing the techniques because it completely disables the service provider's monitoring capability, leaving service providers at great security risk.

Meanwhile, on the contrary, there is a compelling reason for cloud provider to monitor its users: to ensure the security of the whole cloud platform. Concerns of cloud provider come from two sides, one is the VM-to-host attack, and the other is the cross-VM attack. An example of the former one is the "Xen 0wning Trilogy" [1]; and an illustration of the latter one is to make use of Amazon EC2 instances to attack other VMs on the same physical node via the cache side channel [33]. Currently cloud providers has to utilize more and more comprehensive approaches to monitor the platform. For example, with the help of Virtual Machine Introspection (VMI) [24, 26, 28] a cloud provider can look into a virtual machine (VM) and enforce security policies. Sometimes the monitoring method even bridges the semantic gap and reveals more details to cloud provider [9]. Allowing the service provider to look into the memory space of the guest operating systems, and to inspect the processes of the client, obviously may lead to disclosure of the client's private data, especially when users are not aware of where their data are hosted and how they are executed.

Consequently, the mutual distrust between the service provider and client leads to the unavoidable conflict of interest [32]. Service providers have to see the internal world of a client's virtual machine to secure the computing environment to ensure the security of the whole cloud, but this definitely violates the privacy requirement of a client. And the root source of the above conflict is the untrustworthy monitoring and management mechanism. When we seriously consider the role of service provider in cloud computing, we may ask: **does the cloud provider really need to directly monitor and manage the VMs?** We argue that in most situations, the cloud management works are actually conducted on higher layers instead of directly on the physical nodes. As long as the high-level security policies are enforced, it is unnecessary for cloud provider to directly tap into the content of VMs. It is true that under some other circumstances, certain domain knowledge

is required so that cloud provider has to directly scan users' sensitive data . However, we may ask another question: **even if the direct monitoring is required, is it necessary to monitor all the data and processes of the VMs?** Since he full-monitoring scheme has dispelled so many privacy-concerning users, cloud provider should re-consider the trade-off of doing so. From the economics perspective, cloud provider may balance the degree of monitoring to attract more users.

Both the indirect-monitoring and partial-monitoring schemes can't be fulfilled by the current virtualization architecture, where cloud provider has the dominant privilege. It generates the need of a new architecture where:

1. the monitoring work is performed by a neutral component, which is not controlled by the cloud provider;
2. the monitoring functionality should not be cut down;
3. the neutral component must provide the attestation mechanism to be challenged.

Therefore, we propose a neutral cloud computing architecture, NeuCloud, based on the idea of disaggregation of privilege. On each node of NeuCloud, only an unprivileged domain (denoted as "Service Domain") is provided to the cloud provider for cloud management. The original privileged functions are sealed into a "Neutral Domain", which is neither controlled by cloud provider or cloud users. The Neutral Domain is responsible to monitor and operate with the users ("Client Domain") on behalf of the cloud provider by receiving rules and policies from cloud provider through a secure communication channel, which greatly reduces the attack surface. Before using the cloud service, a client should first attest whether NeuCloud is enabled. The attestation will also be carried out during VM migration. In this way, neither the cloud functionality and security nor the users' privacy should compromise. In fact, the "Neutral Domain" can be configured according to a mutual agreement between the client and the cloud provider to avoid any dispute of service.

We implemented NeuCloud Architecture on both Xen, a Type-I hypervisor, and KVM, a Type-II hypervisor. Our experimental study shows that the overhead of NeuCloud is negligible.

In summary, this paper makes the following contributions:

- To the best of our knowledge, NeuCloud is the first effort to enable privacy-preserving monitoring in cloud computing architecture design.
- From economics perspective, the reason to switch to privacy-preserving monitoring and the optimal monitoring degree has been discussed. Although the current full-monitoring mode contributes to the overall security of the cloud environment, but it hurts the privacy of clients. Through partial-monitoring, enabled by NeuCloud, cloud provider may gain more by attracting those users who originally concerned about privacy-leakage.
- We implement NeuCloud architecture on both type-I and type-II virtulization with negligible overhead.
- Our implementation greatly reduces attack surface of the privileged zone. Moreover, little effort is needed for current cloud provider to turn to NeuCloud.

The rest of the paper is organized as follows. In Section 2, we analyse the motivation for cloud provider to switch to privacy-preserving monitoring and what the optimal monitoring degree is. In Section 3, we present the design of NeuCloud where privacy-preserving monitoring is enabled. We also describe the detailed implementation on both type-I and type-II virtualization platform in Section 4. In Section 5, we evaluate both the security and performance of NeuCloud architecture. And some related works are discussed in Section 6.

2. MOTIVATION

An economics-based analysis can be performed upon the privacy-concerned cloud computing market. For simplicity, we assume that all the cloud users share the same resource consuming rate, so that we can use N, the maximum number of users that a cloud platform can host, to describe the total computing resources of one cloud provider; the number of actual users being hosted is denoted as n; p is used to refer to the portion that the cloud provider wants to monitor; $n = D(p)$ measures the number of users with sensitive content but still willing to outsource their data to this cloud provider; $n = S(p)$ measures the number of users the cloud provider is willing to host, given the restraint of p. It is obvious that D is decreasing and S is increasing. Moreover, if n is fixed, p can be derived by evaluating the inverse function $D^{-1}(n)$ or $S^{-1}(n)$. Assume that the utility of cloud provider upon the user population can be measured as $U(n)$, and the utility lost due to the risk of not monitoring the whole content is $V(1-p)$, where U is increasing with n and V is increasing with $1-p$. The total utility of cloud provider is represented by μ.

Cloud providers, just like those currently exist in the market, can monitor the whole content of cloud users; or on the contrary, they can give up this privilege thoroughly; otherwise they can tradeoff by monitoring only part of the content. Correspondingly, p can be a value between 0 and 1. The most popular question upon any security approach is: **whether it is worthy to do so?**. There should be a good reason for cloud providers to switch from full-monitoring to partial-monitoring, otherwise they would prefer the old fashion, which seems more riskless.

Given two monitoring strategies with different monitoring degree, 1 and $0 \leq p_0 < 1$, and the cost due to platform modification due to switching from $p = 1$ to $p = p_0$ is C, the utility difference for cloud provider is:

$$\begin{aligned}\Delta\mu &= U(D(p)) - V(1-p) - (U(D(1)) - V(0)) - C \\ &= U(D(p)) - U(D(1)) - (V(1-p) - V(0)) - C\end{aligned} \tag{1}$$

From the above equation we can see that the full-monitoring scheme is not necessarily always the optimal one. In case that $\Delta\mu > 0$ is satisfied, a rational cloud provider should be willing switch to the privacy-preserving monitoring scheme with the cost C. In reality, if the overall benefit is positive, the cloud provider would always prefer to attract more cloud users with reasonable opportunity cost.

Another question might arise: **even if the cloud provider is willing to switch to the monitoring approach where users' privacy is preserved, what's the optimal monitoring degree?** Actually it depends on the type of cloud computing market. If the market is monopoly, namely there is only one choice for those users with sensitive data but want to outsource their computation, the only cloud provider has full control over the market. In such case, the cloud provider will try to maximize the resource utilization of the cloud platform, so his utility would be:

$$p = D^{-1}(N) \tag{2}$$

$$\mu(N) = U(N) - V(p)) \tag{3}$$

The optimal value of N can be solved by applying the first-order condition:

$$\frac{\partial\mu(N)}{\partial N} = \frac{\partial\mu(N)}{\partial N} - \frac{\partial V(D^{-1}(N))}{\partial N} \tag{4}$$

Using the optimal N' calculated from Equation (4) , the cloud provider can deduce the optimal monitoring portion p' according to Equation (2).

However, if the market is competitive, where there are lots of homogeneous cloud providers, the optimal monitoring degree $p^\star$ is determined by the market equilibrium:

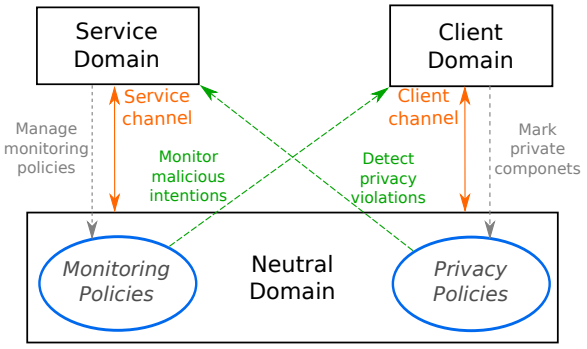


Figure 1: The illustration of the relationship between Service Domain, Client Domain and Neutral Domain. Neutral Domain helps service provider to perform the monitoring and management work, and detects privacy violations on behalf of clients.

$$D(p^\star) = S(p^\star) \tag{5}$$

$$\mu(n) = U(S(p^\star)) - V(p^\star)) \tag{6}$$

Finally, if the situation is oligopoly, which lies between full competition and no competition (monopoly), the cloud providers decide their monitoring strategies independently and simultaneously. Assume that there are two homogeneous cloud providers. The users who are willing to use the service from the first cloud provider is $D_1(p_1, p_2)$, which is a decreasing function of p_1 and an increasing function of p_2. The case for the second cloud provider is similar. Generally the utility of the ith cloud provider is given by:

$$\mu_i(p_i) = U(D_i(p_i, p_{-i})) - V(p_i) \tag{7}$$

where p_{-i} denotes the monitoring degree of all cloud providers other than the ith one.

Then given the other cloud providers' strategies, the optimal monitoring degree can be solved through first-order conditions:

$$\frac{\partial \mu_i(p_i)}{\partial p_i} = \frac{\partial U(D_i(p_i, p_{-i}))}{\partial p_i} - \frac{\partial V(p_i)}{\partial p_i} \tag{8}$$

To summarize, if there exists a cloud architecture, with privacy-preserving monitoring enabled and with the modification cost of C satisfying that the $\Delta\mu$ in Equation (1) is positive, the cloud provider would be willing to switch to this cloud architecture. And the optimal monitoring degree is determined depending on the market type.

3. NEUCLOUD OVERVIEW

The NeuCloud archtecture is illustrated in Figure 1, and the detailed description of each domain can be found in Table 1. In the traditional cloud computing architecture, serving as the intermediate between client domains and hardware, the cloud provider has the dominant privilege. However, in the model of NeuCloud, the cloud provider is moved out of the privileged intermediate layer into a specific Service Domain, and the Neutral Domain takes over the central-control work. In the NeuCloud's symmetric design shown in Figure 1, neither cloud provider nor client is more privileged. In fact, there is no particular difference between Service Domain and Client Domain, except the accessibility to different interfaces provided by Neutral Domain. To manage the computing node, cloud provider can send service request, update security policies and gather monitoring results from the Service Domain; while cloud users can maintain their privacy policies, receive privacy violation reports and attest the authenticity of the platform from the Client Domain.

Cloud management and monitoring: as shown in Figure 1, the Service Domain can send service requests coming from Cluster Controller or Cloud Controller (CC/CLC) to Neutral Domain. With the deprivation of authority from cloud provider, one may concern that whether the administrative functionalities are restricted. Actually all the privileged executions, like domains/VMs management (for example "Create VM", "Launch VM" or "Migrate VM") and platform configurations (for example those system parameters that should be adjusted according to hardware specifications), can be performed by Neutral Domain on behalf of the Service Domain . A monitoring policy database is also set up for the Service Domain to store security policies or signatures to identify malicious intentions. With a higher privilege, the Neutral Domain is responsible to execute the service requests received and monitor all the domains/VMs on the node, including both the Service Domain and the Client Domain, according to the security policies. Those monitoring objects will be extracted from the monitoring policies, and are reported to the corresponding clients. Because now a cloud client can make sure what part of his content is under monitoring and what is not, cloud provider should balance the monitoring degree as what has been discussed in Section 2. Except for revealing the monitoring targets to cloud users, Neutral Domain will also check if user-defined privacy policies are violated, which will be introduced in the following part.

The first advantage of involving Neutral Domain in is that the monitoring process is no longer stealthy to cloud users; the second is that the Neutral Domain makes it possible for the two parities to reach an agreement on monitoring degree, because it can neutrally prove the fact of partial-monitoring to the cloud users.

Privacy protection: Cloud Domain is the zone where users' data and computations are hosted. Instead of the "pure guest mode", now cloud users are endowed with more privileges. They can maintain privacy policy databases (one for each domain/VM) that describes which data or process is private and cannot be directly seen by cloud provider. During the monitoring procedure on behalf of the Service Domain, the Neutral Domain will verify whether the content being monitored matches the description in privacy policies. If so, a conflict happens. A client cannot directly touch the Neutral Domain, but with the help of vTPM mechanism [6], he is able to challenge and attest whether the data and computations are held on the NeuCloud platform.

To solve the monitoring/privacy conflict: there exists a security-policy database for Neutral Domain to refer to in order to verify malicious content; and at the same time Neutral Domain is also responsible for protecting the privacy of Client Domain. Given a piece of a Client Domain's code or data X, a monitoring policy m, and a privacy policy p, we denote $X = m$ if X matches m (suspected as malicious) and $X = p$ if X matches p (marked as privacy). There might be four situations:

1. $X \neq m$ *and* $X \neq p$
2. $X = m$ *and* $X \neq p$
3. $X \neq m$ *and* $X = p$
4. $X = m$ *and* $X = p$

The first three situations are easy for the cloud provider and client to reach an agreement. But a conflict may happen in situation 4. In such cases, which one should the Neutral Domain yield to, cloud provider or cloud users? Simply showing partiality for either of them would make the Neutral Domain meaningless, for example:

1. cloud provider is benign and a client is malicious; and the client tries to hide his behaviours by marking his malicious code as private. So the Neutral Domain cannot simply yield to cloud

Table 1: Detailed description of each domain in NeuCloud.

Domain Name:	Client Domain	Service Domain	Neutral Domain
Description:	The collection of all the client VMs on the physical node	The component directly controlled by cloud provider	The root complex
Components(KVM):	Client VMs	A special VM opened to cloud provider, and with device pass-through	Host OS with KVM module
Components (Xen):	Domain 2,3,...,N	A customized Domain 1 with device pass-through	Reduced & sealed Domain 0 and Xen hypervisor
Privileges:	Maintain privacy policies	Provide device drivers Communicate with Cluster Controller (CC) Pass service requests to Neutral Domain Provide security policies Maintain other libraries (etc. patches)	Provide service interface to Service Domain Create and delete Domains/VMs Launch and stop Domains/VMs Migrate Domains/VMs Monitor Domains/VMs Patch/update Domains/VMs

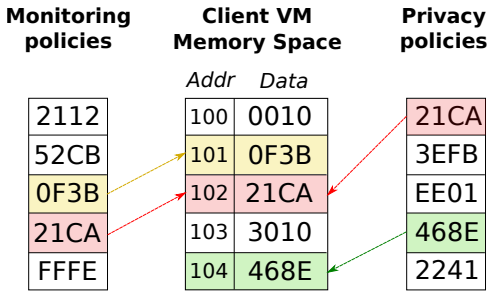


Figure 2: The situation where conflict between monitoring and privacy protection happens.

users.

2. Client is benign and cloud provider is malicious. In this case, the cloud provider deliberately uses monitoring policies to infer whether a Client Domain has certain processes or data. For example, the cloud provider may insert "issue of shares" into policies to try and infer commercial secrets. So the Neutral Domain cannot simply yield to cloud provider, either.

To solve this, on one hand, to guarantee the security of the cloud environment, Neutral Domain must warn a cloud user about his security policy violation. Neutral Domain will prevent the client from using suspected data and suspend suspected process. When security policy violation detected, Neutral Domain will let the client make a decision - if the client is innocent, he can argue with cloud provider or move his business to other cloud providers; if the client agrees to sanitize his code, Neutral Domain will resume the program status and execute the sanitized code.

And on the other hand, if policies are violated by a client, the cloud provider should not receive any messages if the violating content is specified in the client's privacy database, aiming at preventing inference attacks from cloud provider. We note that the cloud users may mark all of their content as private to prevent any information flowing to cloud provider, but still this cannot hide their malicious activities due to the presence of monitoring policies defined by cloud provider.

A certain example may help to make this problem more tangible. As depicted in figure 2, Neutral Domain is monitoring a Client Domain's memory space according to the cloud provider's policies and the client's privacy definitions. The client's data in address 101 is suspected as malicious and not marked as private, so Neutral Domain will notify both cloud provider and client that policy $0F3B$ is violated by memory address 101 of this VM. Both parties will take actions accordingly. The client's code in address 102 is matched by both secure policy and privacy policy, in this case, then, Neutral Domain will suspend the process related to this memory content and require the client to sanitize his program. In this case, Neutral Domain will not inform cloud provider because that would violate the privacy policy.

NeuCloud attestation: In Appendix A, we provide a brief introduction to Trusted Platform Module [23]. Currently, most servers have TPM chips installed by the manufacture. With the aid of a TPM, a client can communicate with a trusted external party, Privacy CA, at any time to verify the certification corresponding to the TPM of the cloud platform. We should note that there is a an important distinction between an "external party" and the "third party" described in the very beginning. The former one is not able to directly touch the data and process on cloud computing platform while the latter one is able to do so since it is involved as an entity (for example as a monitoring entity) as well. The Privacy CA maintains a database of all the certified TPM chips. During the very beginning, the client should initially verify whether the computing node provided by cloud provider is built up as the one described later. Therefore, with the help of TPM, the cloud provider can boot the Neutral Domain and guaranteed its integrity.

4. NEUCLOUD DESIGN

4.1 Goals and assumptions

Scope. The scope of this paper is limited within the architecture-level design and implementation of a neutral cloud computing platform. Therefore, we will not go deep into the details of low level or peripheral security mechanisms. What's more, some important functionalities in cloud computing, for example patching, are convenient to be implemented in NeuCloud, but as space is limited we will cover them in other papers.

Goals. The design of the neutral cloud computing platform, NeuCloud, should meet the following goals. 1) The Neutral Domain should be *neutral*. Both the cloud provider and the cloud provider are treated impartially. 2) The costs-effectiveness of NeuCloud should be economically sound. For example, the design should be easily integrated into the current cloud architecture with both Type-I and Type-II virtulization. 3) Monitoring capability should be enabled while preserving privacy. 4) The platform should be verifiable during the life cycle of service. 5) All cloud management functions, such as VM migration, should be enabled. 6) The performance overhead should be negligible.

Assumptions. First, we assume that the root complex, namely the Neutral Domain, described in this paper is secured, for example by using HyperSafe approach [38]. HyperSafe includes two key techniques, "non-bypassable memory lock down" and "restricted pointer indexing", to reliably provide the integrity guarantee with a small performance overhead. HyperSafe can not only guarantee the load-time

integrity of the hypervisor but also maintains the same level of integrity continuously throughout the lifetime of the hypervisor. Moreover, it does not require any effort from the cloud provider to modify hardware and software, thus avoiding the involvement of the cloud provider in the attestation chain.

Second, we also assume that the cloud provider is rational. Although with the involvement of the Neutral Domain we can prevent the cloud provider's Service Domain from arbitrarily touching users' privacy, it is not guaranteed that some cloud provider may circumvent the software architecture and eavesdrop through physical layer. However, if the cloud provider wants to physically eavesdrop the memory bus, he needs special hardware facilities (a dedicated motherboard for example). Even though the cost brought in by this additional hardware might be negligible, such actions would leave behind hard evidence pointing to this privacy violation. Basically we assume the cloud provider won't take this risk.

Third, the target platform is assumed to be equipped with TCG's trusted boot hardware, especially the TPM chip, which is the core component to authenticate hardware devices and all the software components running above. Most of the common motherboards have already been equipped with TPM chip. With the support of TPM, a client can challenge the authenticity of the root complex hosting his data and computations. These processes are all independent of the involvement of the cloud provider, and as a result avoid any attempts by the cloud provider to secretly tamper with the platform or Client Domain.

4.2 Implementation

NeuCloud is a universal model that can be implemented on both type-I and type-II virtualization. In this section, we describe how to implement NeuCloud's computing node for both types of virtualization. The platform specification is attached in Table 3 of Appendix B, of which the TPM chip and the CPU featured with Intel VT-d technology are specifically required.

Neutral Domain. The Xen based Neutral Domain (Figure 3a) contains the sealed and reduced Domain 0 as well as the Xen hypervisor itself. There is no special modification to be made with Xen VMM (we have assumed that it is pre-configured with "HyperSafe" protection). But the original Domain 0 should be modified by: first, all the peripheral drivers (e.g. network card driver, USB driver) are removed, as these drivers are no longer needed and we can reduce the code volume of Domain 0 to minimize the surface exposed to attackers; second, we remove all the Linux user profiles and block all network ports, prohibiting both local login and net login, of which the purpose is to make the sealed Domain 0 a "closed-box" that cannot be logged into locally or remotely; third, we implement an agent to watch the communication channel between sealed Domain 0 and Service Domain (Domain 1), which is responsible for transferring service requests; finally, we equip the sealed Domain 0 with XenAccess to monitor other domains.

The KVM based Neutral Domain (Figure 3b) consists of the host OS and the KVM module. All the peripheral device drivers are eliminated in the host OS and are provided by the Service VM instead. The host OS should also be modified by removing all the login profiles, blocking all the internet ports, and equipped with VM intrusion detection tools.

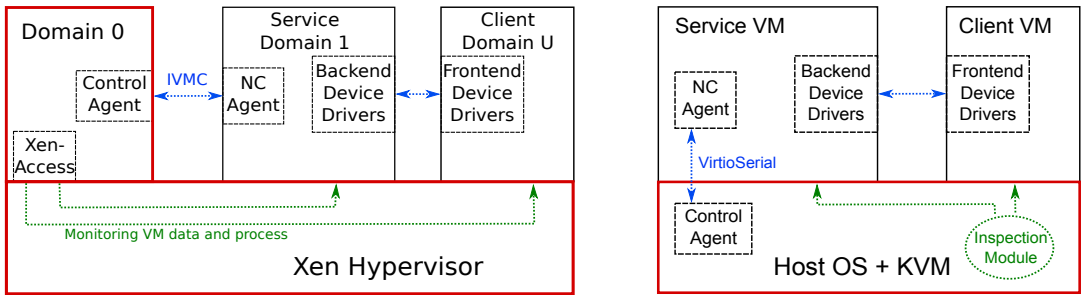
Service Domain. For Xen-based platform, a pre-configured Domain 1 will be automatically launched as the Service Domain, along with the boot of sealed Neutral Domain 0. This can be achieved by moving the configuration file of domain-1 under /etc/xen/auto. The operation system of Domain 1 in our implementation is currently a para-virtualized linux (openSUSE 11.3, x86_64, with 2.6.34.7-0.7-xen kernel), but the full-virtualized linux is also applicable, as long as the kernel is compatible with the Intel VT-d feature. Two modifications are needed for this Service Domain: first, we write a Node Controller (NC) agent to communicate with the CLC/CC. On receiving the directions from the CLC/CC, the NC agent will translate them into requests acceptable by sealed Domain 0, and send these requests through the communication channel to the sealed Neutral Domain; second, those drivers removed from the sealed Domain 0 should be implemented in the Service Domain. The passthrough of those devices to the Service Domain can be achieved by adding the physical address of the devices into the configuration file of the Service Domain, for example: "`pci=['0000:0b:00.0','0000:0b:00.1']`". Once the Xen backend device drivers are loaded in Service Domain and the Xen frontend device drivers are loaded in client VM, the client can get support from these devices.

As for a KVM-based platform, the situation is similar. The Service Domain is automatically booted along with the host OS. And all the devices assigned to be managed by the Service Domain have to be claimed in the Service Domain's configuration file, like "`-device pci-assign, host=0b:00.0`".

Trusted Management Communication channel. As mentioned above, on a Xen-based platform, there should be a trusted inter-VM communication (IVMC) channel connecting the Service Domain 1 and the Neutral Domain 0. We implement this channel on a Xen-based Neutral Domain with the help of XenBus [4] and XenStore [20], as illustrated in Figure 4a. XenStore is designed as a hierarchical collection of key-value pairs and XenBus provides a bus abstraction to communicate with these key-values (or entries). Each domain has a directory hierarchy containing data related to its configuration. It is permitted for domains to modify, create and delete information in their own special sub-trees in XenStore, but they are not allowed to access other domains' sub-trees. The Neutral Domain 0 is able to read from and write to any key-value in XenStore. Due to its VMM-based access control feature, this inter-domain communication is secure. In our architecture, the Neutral Domain 0 places a "watcher" on a certain entry in the Service Domain's XenStore directory through XenBus. Whenever the Service Domain writes to this entry, the Neutral Domain 0 can immediately detect the change and respond to the request. Note that only Service Domains and the Neutral Domain 0 have access to the Service Domain's entries, so that a client's domains are unable to forge a cloud platform request. At the same time, the cloud provider is unprivileged to access other domains' content in XenStore.

To provide a secured communication channel between the Service VM and host OS for KVM-based platform, a character device is specially attached to the Service VM and a corresponding unix socket is created in host OS, as shown in Figure 4b. This two-way stream socket communication takes the advantage of the VirtioSerial feature of KVM [3]. For example, in our implementation, "`-device virtio-serial -chardev socket, path=/tmp/foo, server, nowait, id=foo`" is added into the configuration file of Service VM. This creates a character device, "`/dev/vport0p1`", in the Service VM and a corresponding Unix socket file, "`/tmp/foo`", inside the host OS. Thus the host OS can communicate through this virtual channel with the Service VM.

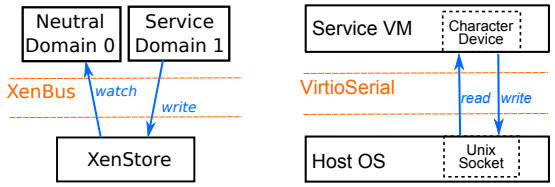
Monitoring and privacy policies and reports. The Neutral Domain should provide a security policy database to the cloud provider and a privacy policy database to each cloud user. Because the volume of the policies and the logs might be as large as a few gigabytes, these data cannot be kept in a shared memory region. Thus we create an image file mounted by both the Neutral Domain and the Service Domain, which contains the security policies; meanwhile, each cloud user can mark its privacy zone in the image file mounted by itself and the Neutral Domain. The monitoring and privacy policies look like:



(a) NeuCloud node implementation on Xen hypervisor

(b) NeuCloud node implementation on KVM platform

Figure 3: NeuCloud node implementation



(a) IVMC on Xen-based NeuCloud

(b) VirtioSerial on KVM-based NeuCloud

Figure 4: Trusted management communication channels of Xen-based NeuCloud and KVM-based NeuCloud.

$$Malicious(ProcessP, DomainU) :=$$
$$Running(ProcessP, DomainU) \wedge Match(ProcessP, MaliciousSigratureM)$$

$$Private(DataD, DomainU) :=$$
$$Read(DomainU, DomainV) \wedge Write(DomainU)$$

The Neutral Domain will gather the information of all the data and processes under monitoring, reporting it to the corresponding client domain/VM. And the Neutral Domain will take actions if the monitoring or privacy policies are triggered or violated.

Initialization of the computing node. Since the Neutral Domain 0 or the Neutral host OS is permanently sealed at runtime, it seems impossible to initialize the platform. The Neutral Domain cannot know in advance about what kind of peripheral hardware should be passthroughed into the Service Domain, and it seems that the cloud provider will never have the opportunity to touch the root system configurations. However, the sealed Domain 0 or host OS is runtime sealed only in the sense of denial of login. Through the communication channel, some configuration requests can be sent by the cloud provider to initialize the Neutral Domain from the Service Domain.

4.3 NeuCloud protocols

It should be emphasized that in spite of the special architecture of Neutral Domain, there is not much difference at first glance between the client and cloud provider. Especially for client, the platform transformation due to moving to a Neutral Domain based cloud will be effortless. Since the Neutral Domain provides management interfaces to the Service Domain, and the network card driver (thus the network controller) is also moved into the Service Domain, the cloud provider can still use NC agent to communicate to upper level CLC/CC and conduct the scheduling. Despite the concerns about the cost of a platform transformation, a client is more likely to be concerned about how to guarantee his security and privacy.

In this section, we describe how to support cloud management with Neutral Domains. To avoid the direct involvement of Privacy CA into cloud protocol, and to improve the performance, we adopt Direct Anonymous Attestation (DAA) [13]. The DAA scheme involves three roles of entities: issuers, signers and verifiers. The issuer, namely the Privacy CA, attests and issues a DAA credential to a legitimate signer; the signer, namely the Neutral Domain, can prove himself to verifier (the client) his DAA signature from Privacy CA. DAA scheme does not require Neutral Domain to provide detailed identity to the client and the client does not need to send the identity to Privacy CA for attestation each time.

In the following discussions we will describe cloud instance initialization, runtime verification and the secure migration of Client Domain. The notation used in this paper is as follows:

$\{msg1, msg2\}$: A combination of two messages.

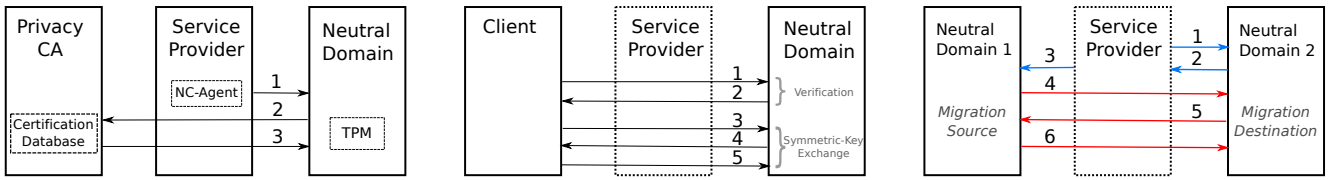
$\{msg\}K_y^x$: The message is encrypted by a public key or signed by a private key, where K is the key name, x ("pub" or "pri") represents the asymmetric key type (public or private), and y indicates who generates this key. Specifically, the endorsed key of TPM is represented as EK_y^x, and the application identity key is represented as AIK_y^x. If neither "pub" nor "pri" appears, then K is a symmetric key.

$A \rightarrow B : \{msg\}K_A^{pri}$: A sends a message to B, with the content encrypted (signed) using A's private key.

n_x: A unique number generated by x, helps to detect message replays.

4.3.1 Adding a new Neutral Domain to the cloud

For platform authentication, Neutral Domain should provide its TPM PCR values which contain the measurement result of critical system components. To certify the PCR values provided to client, Neutral Domain needs to sign them using its Application Identity Keys (AIKs). But when a client is verifying the authentication of PCR values, how can he tell that the signing key is actually from a authenticated TPM? We need a Privacy CA, a trusted external party, to sign the AIKs belong to a authenticated TPM. Privacy CA holds a database of all the public EKs (which is unique to each TPM chip and the private part is permanently sealed in the chip) from TPM chip manufactures. We follow a DAA-join process to add new Neutral Domains to the cloud. During DAA-join process, cloud provider send a request to Neutral Domain to ask for the public EK, public AIK and a public RSA key (which will be used for data encryption, as AIK and EK cannot directly be involved in arbitrary data encryption), and deliver them to Privacy



(a) Adding new Neutral Domain (b) Neutral Domain verification (c) Trusted VM migration

Figure 5: NeuCloud management

CA. If the public EK is a valid one in the database, Privacy CA will sign the public AIK and public RSA key. Then whenever Neutral Domain shows the keys with Privacy CA's signature, the client can trust the keys. The protocol, as shown in Figure 5a, is described as following:

1. Any server with Neutral Domain should be certified before it is enrolled as a cloud computing node. So cloud provider, through Service Domain, sends a request to the Neutral Domain for the public AIK, public EK and a public RSA key:
   $cloudprovider \rightarrow NeutralDomain : n_{sp}$
2. Neutral Domain responses to the request, sending those keys to Privacy CA (via cloud provider):
   $NeutralDomain \rightarrow PrivacyCA :$
   $\{AIK^{pub}_{NeutralDomain}, EK^{pub}_{NeutralDomain}, K^{pub}_{NeutralDomain}\}$
3. If the public EK matches a valid entry in its database, Privacy CA will return with the signed public AIK and RSA public key, which will be further used by Neutral Domain to attest itself to client:
   $PrivacyCA \rightarrow NeutralDomain :$
   $\{\{AIK^{pub}_{NeutralDomain}\}K^{pri}_{CA}, \{K^{pub}_{NeutralDomain}\}K^{pri}_{CA}\}$

As the messages delivered in this protocol only contains public keys, Neutral Domain does not need to encrypt them to prevent cloud provider to see them. But what if cloud provider use the keys signed by Privacy CA on a non-Neutral Domain platform? In this situation, client may be deceived to trust the non-Neutral Domain platform, but the cloud provider still cannot see the data of client encrypted by these public keys since the private keys are all handled by Neutral Domain's TPM only.

4.3.2 Neutral Domain verification

In cloud computing environment, client should firstly register to cloud provider to obtain an identification. Only customers with identifications are permitted to use cloud computing resources and cloud provider will associate the service fee to each client. After the registration, client is allowed to upload his own image to the computing node. In this step, client may concern about whether the computing node is a secured one with trusted computing architecture. Therefore, we provide the DAA-verify scheme for client to challenge the Neutral Domain based computing node in the cloud.

With the signature from Privacy CA described in Section 4.3.1, client can verify the public keys from an Neutral Domain and trust data signed with these public keys. We use a DAA-verify protocol to verify an Neutral Domain. The DAA-verify protocol contains two phases (Figure 5b): the first phase verifies whether the target platform is Neutral Domain-based, and the second phase exchanges symmetric session key after client establish the trust relationship with target platform. Although all the communications between Neutral Domain and client are transferred by cloud provider, the cloud provider can only learn the public keys or cipher text. The verification procedure is described as follows.

1. If a client have registered to cloud provider for cloud computing service, cloud provider will allow this client to send request to Neutral Domain. During the greeting step, the client sends a request to Neutral Domain for remote authentication, in a form that Neutral Domain can ensure this is a registered and valid client:
   $Client \rightarrow NeutralDomain : n_{client}$
2. When the Neutral Domain receives the request, it will check its public key database - if this is not a existed client, a new entry to store public key and session key of the client will be created. Then Neutral Domain replies with certified public AIK and public RSA key, and PCR values signed with AIK:
   $NeutralDomain \rightarrow Client :$
   $\{\{AIK^{pub}_{NeutralDomain}\}K^{pri}_{CA}, \{K^{pub}_{NeutralDomain}\}K^{pri}_{CA},$
   $\{PCR_{NeutralDomain}\}AIK^{pri}_{NeutralDomain}\}$
3. The client then verify the signature of Privacy CA. Once the TPM authentication is successfully verified, the client use the public AIK to further verify the signed PCR values, to attest whether the target platform is based on Neutral Domain. If passed, client can use the trusted public RSA key to encrypt a symmetric session key and his public key, sending them to Neutral Domain. The reason why we create a symmetric key is because using symmetric key is more efficient than using asymmetric key to transfer large files; the public key of client will be used to verify signature on security policy libraries (as we have described in 4, if a client agrees with the policies provided by cloud provider, he should sign the policy library. Then Neutral Domain is able to verify whether client and cloud provider have reached an agreement):
   $Client \rightarrow NeutralDomain : \{K_{client}, K^{pub}_{client}\}K^{pub}_{NeutralDomain}$
4. Neutral Domain sends a response, if the keys above is received:
   $NeutralDomain \rightarrow Client : \{n_{NeutralDomain}\}K^{pub}_{client}$
5. Client upload the VM image encrypted by the session key:
   $Client \rightarrow NeutralDomain : \{VM_{client}\}K_{client}$

Then Neutral Domain can decrypt and register the image if the "IMAGE REGISTER" direction is passed from cloud provider. Note that only Neutral Domain is able to decrypt the client's image and once uploaded, cloud provider, residing in Service Domain, has no way to access the content of client's image even it is decrypted. To further guarantee the security of client's VM, client can launch the runtime attestation of the computing platform at any time. As the client's VM is now directly launched on Neutral Domain, it is able to challenge the bootstrap record through vTPM at will.

4.3.3 Trusted virtual machine migration

Migration is necessary for a cloud computing platform for performance considerations. But with the power to migrate client's VM, cloud provider is able to move (or copy) client's VM to a non-Neutral Domain platform and look inside. Of course, we cannot rely on the client himself to verify the computing platform through every second, because it will bring in high overload. Thus the NeuCloud should take over the responsibility. Suppose the migration source is Neutral Do-

main_1 and destination is Neutral Domain_2, as shown in Figure 5c. To secure this operation, the protocol is as follows:

1. cloud provider checks with Neutral Domain_2 to see whether it has enough resources to move a new VM in. If so, cloud provider sends a notification to Neutral Domain_2. cloud provider has to sign the notification, because the message is not oriented from Service Domain on Neutral Domain_2 but from CLC/CC instead. A signed notification can prevent malicious party from migrating client's VM among Neutral Domain-based platforms. Although the migration from Neutral Domain-based platform to another won't lead to privacy leakage, but if this happens without cloud provider's awareness, cloud provider cannot locate this client's VM in the future. This is a typical denial-of-service (DOS) attack.
   $cloudprovider \rightarrow NeutralDomain\_2$:
   $\{n_{cloudprovider}\}K^{pri}_{cloudprovider}$
2. Neutral Domain_2 returns the certified public AIK and public RSA key, along with AIK signed PCR values. As we have discussed, those who received the public keys signed by Privacy CA can trust the authentication of the owner of these public keys. And because these are public keys, there is no need to encrypt them:
   $NeutralDomain\_2 \rightarrow cloudprovider$:
   $\{\{AIK^{pub}_{NeutralDomain\_2}\}K^{pri}_{CA}, \{K^{pub}_{NeutralDomain\_2}\}K^{pri}_{CA},$
   $\{PCR_{NeutralDomain\_2}\}AIK^{pri}_{NeutralDomain\_2}\}$
3. cloud provider then transfers these data to Neutral Domain_1, which should also be signed by cloud provider to prevent other malicious parties to launch the DOS attack:
   $cloudprovider \rightarrow NeutralDomain\_1$:
   $\{n'_{cloudprovider}, \{AIK^{pub}_{NeutralDomain\_2}\}K^{pri}_{CA}, \{K^{pub}_{NeutralDomain\_2}\}K^{pri}_{CA},$
   $\{PCR_{NeutralDomain\_2}\}AIK^{pri}_{NeutralDomain\_2}\}K^{pri}_{cloudprovider}$
4. The Neutral Domain_1 then verifies the platform information of Neutral Domain_2. If passed, Neutral Domain_1 sends a symmetric session key to Neutral Domain_2, via cloud provider. As we have mentioned, this is because using symmetric key is more efficient than using asymmetric key to transfer large files:
   $NeutralDomain\_1 \rightarrow NeutralDomain\_2$:
   $\{K_{NeutralDomain\_1}\}K^{pub}_{NeutralDomain\_2}$
5. On receiving the above message, Neutral Domain_2 informs Neutral Domain_1 to start migration:
   $NeutralDomain\_2 \rightarrow NeutralDomain\_1$:
   $\{START\}K_{NeutralDomain\_1}$
6. And migration securely launched, with the content of Client Domain and its hash value transferred:
   $NeutralDomain\_1 \rightarrow NeutralDomain\_2$:
   $\{VM_{client}, H_{client}\}K_{NeutralDomain\_1}$

5. EVALUATION

5.1 Economics analysis

Assume the number of users with sensitive data and computation and willing to move on cloud computing has a linear relationship with the monitoring degree. The D function introduced in Section 2 can be written as:

$$D(p) = D(1) + \lambda(1-p), where \lambda > 0$$

And we also assume that the cloud provider's utility is in a linear form:

$$U(n) = \theta n$$

$$V(1-p) = \eta(1-p)$$

So the Equation (1) can be transformed to:

$$\Delta\mu = (\theta\lambda - \eta)(1-p) - C$$

The optimal $p^\star$ can be calculated according to the discussion in Section (2), depending on different market type. So the cloud provider can decide whether moving to NeuCloud architecture is worthwhile, by measuring if the following condition is satisfied:

$$C > (\theta\lambda - \eta)(1-p^\star)$$

5.2 Security analysis

We have assumed that the root complex, namely the Neutral Domain, is secured, for example by using HyperSafe approach [38]. And by removing drivers from the Neutral Domain and sealing it, the attack surface of the privilege zone is largely reduced. So the vulnerabilities due to the compromised Neutral Domain is eliminated. In the following part, we analyze how the following threats are handled by the NeuCloud architecture design.

Attacks from the cloud provider. In these attacks, the cloud provider or a disgruntled employee, may want to access the cloud users' data and processes, violating the privacy requirements. In the NeuCloud architecture, the cloud provider no longer has the privilege required to launch such operations. Being isolated into a unprivileged zone same as any clients, the cloud provider is unable to view the status of a client domain/VM's virtual CPU, to arbitrarily examine a client domain/VM's memory, or to gain access to an image or secondary storage that is not assigned to him. All the service requests have to be sent to the Neutral Domain to get executed, where the privacy violations would be detected.

Attacks from cloud clients. There are two possibilities for this kind of attacks: maybe the clients themselves are malicious; or their domains/VMs have been taken over by malicious parties. But because all the clients' data and processes are under monitoring of the Neutral Domain, according to the security policy database maintained by the cloud provider, part of the attacks will be detected immediately. The cloud provider is responsible for providing the rules to be referred to in attack detection and forensic analysis. It is true that the security policies cannot cannot cover all the possible malicious behaviours, but the design of security policies is not within the scope of this paper. Our effort is to provide the neutral platform where service provider and cloud users can easily reach an agreement on monitoring.

5.3 Performance assessments

The performance influence of the final platform comes from two sides. Apart from those brought in by our architecuture, there exist overheads from the "assumed" components introduced in others' works that we depend on, like HyperSafe. The latter one mainly influences the hypervisor, while our modification is upon a higher architecture level. Although the functional integrity of NeuCloud relies on the low-level mechanisms, the performance is independent from them. So here we only consider if NeuCloud alone brings in negligible overhead. The specification of our test environment is in Table 3 of Appendix B. One possible bottleneck of performance is the communication channel between the Service Domain and Neutral Domain. Take a KVM-based NeuCloud computing node as an example. The data transfer rate between the Service Domain and the Neutral Domain is shown in Figure 6, from which we can calculate that the transfer rate is around 100MB/s transferring files around 10kB and it slightly drops to 81MB/s transferring files around 100kB. Since in our current design the largest message transferred through this channel (the signed PCR values) is smaller than 5kB, this delay can be ignored.

Another concern about performance is that whether NeuCloud can

Table 2: Performance measurement

Process	Delay(ms)
DAA verification	41.967
Migration	124.817

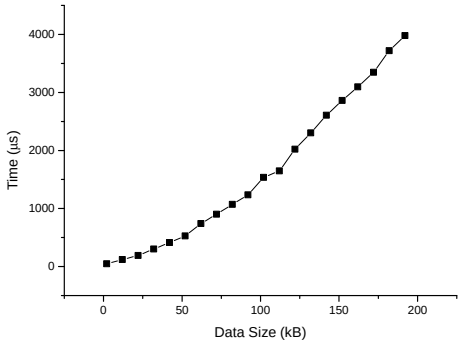


Figure 6: Data transfer rate of service VM to host OS on KVM-based NTCB. The X-axis represents for data size and the Y-axis represents the time cost in transfer.

keep the same hardware efficiency as the ordinary platforms, among which the network transfer capability of the Service Domain is the most important. As we move the NC agent as well as the network card driver into the unprivileged Service Domain, it is possible that the network transfer performance will be influenced. A client may be concerned about this performance cut-down because it may result in a slower network response and a slower image upload or migration speed. The experimental results of network transfer rates are shown in Figure 7. In our results, the transfer rates are almost the same.

Apart from efficiency, stability is also very important. The network delay time distributions of different platforms are shown in Figure 8, which indicates that the delay time of NeuCloud is even more concentrated. This definitely helps to improve the customer experience in cloud computing environment. The cause of the improvement is easy to infer. By outsourcing service tasks and communication works to Service Domain, Neutral Domain can only focus on some certain tasks, leading to better performance than the ordinary cloud platform where the cloud provider's VM has to take all the above responsibilities.

We can also conclude that the client's experience will not be affected under NeuCloud, because moving to NeuCloud requires no modification upon client's domain/VM. The only difference is that the device driver is provided by Service Domain instead of the root complex. Given that the driver supporting mechanism has not been changed, and the hardware efficiency and stability remain the same, the client should not perceive any difference in performance from that of the ordinary architecture.

As for the communication interface of the whole cloud, the typical delay from the client-side view is shown in Table 2. Before client can upload image and register for an instance, the client should spend time on DAA verification of the NTCB. This only costs less than 50ms (in our experiment, client and cloud provider is in the same local-area-network). And the delay from the issuing of "VM MIGRATION" to its launch is less than 150ms. The experimental results indicate again that NeuCloud will not bring in perceivable influence to client experience.

6. RELATED WORK

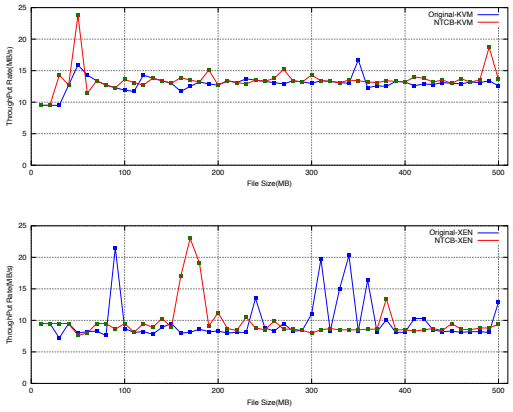
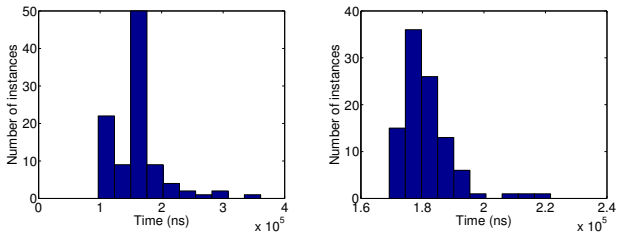
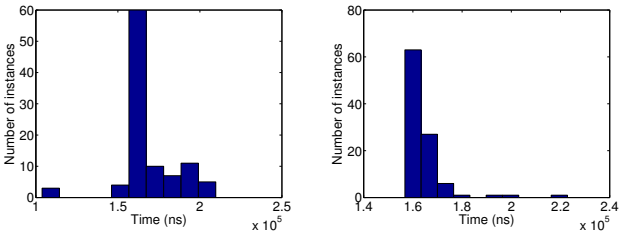


Figure 7: A comparison of network transfer performance of client VM between Opennebula and NeuCloud.



(a) Delay distribution of ordinary KVM platform

(b) Delay distribution of ordinary Xen platform



(c) Delay distribution of KVM-based NeuCloud

(d) Delay distribution of Xen-based NeuCloud

Figure 8: A comparison of TCP/IP response delay distribution of different platforms.

To benefit the service provider's interests and prevent client from using cloud services to host Crimewares [8] , the Intrusion Detection System (IDS) [24] and Virtual Machine Introspection (VMI) [26, 28] may be deployed. These techniques can summarize the state of a VM as the sum of its software state (including the content of both physical memory and hard disk) and hardware state [31]. In fact, there have been plethora work discussing about how to better monitor VMs [15, 27, 30, 14]. These techniques greatly enhance the monitoring capability but none of them take privacy issues into consideration.

The protection of a client's private data from an intrusive service provider has become a hot-spot in recent years. There are numerous approaches focusing on architecture modifications to protect a client's privacy from the service provider. Those efforts, according to their mechanisms, can be classified into three categories: (1) cut-

ting off some of the privileged components of the service provider (represented by [34, 10, 19]), or (2) moving some of the privileged functions out of the service provider's direct control (represented by [25]), and (3) integrating the whole virtualization functionality into the processor and relying on the tamper-resistance of hardware to protect client VM from service provider.

The"root secure" property, introduced in the IBM secure coprocessor's approach [34] and in Terra architecture [10], can be considered the pioneer of privacy protection from the service provider on the architecture level. The phrase "root secure" means that data stored on a cloud platform is secure from examination and tampering even by the platform owner who has root level access. It opens up a research direction, to provide cloud computing privacy by dividing the TCB and excluding those privileged, privacy-threatening components out of the TCB. As a successor to this kind of approach, [19] is able to secure the client's VM by intercepting and restricting the hypercalls made from Dom 0 to Dom U and introducing some new hypercalls. In all of these approaches, a client's privacy is perfectly protected, but still, none of them endows monitoring power to the cloud computing platform. Besides, these approaches heavily depend on complicated functions provided by the privileged domain, which are hard and impractical to implement in industry.

The second category of approaches, like [25], involve disaggregating the management components from the service components, which is similar to the idea of this paper except that they still keep the service provider in the privileged domain. In [25], part of the important privileged components, the domain builder, is moved out of the service domain into a minimal trusted compartment. This disaggregation mechanism further improves upon the thought of dividing the TCB. But in [25], the location of the service provider in the privileged domain still forces the cut-off of some important control functionalities needed by cloud computing. For example "foreign mapping", which is required during migration of a client's VMs, has to be removed, and thus some of the cloud computing functionalities are deprived. And again, the security monitoring capability is weakened.

The third one, to thoroughly eliminate Hypervisors and rely on the tamper-resistant processor to provide virtualization, as introduced by "NoHype" [17], is probably the ultimate scheme to defend malicious or compromised service provider. However, there is still pretty much effort ahead to make it practical. As the "NoHype" itself, the requirement that each VM should occupy a unique core is still unacceptable to most service providers nowadays running on common commercial CPUs.

Apart from all the above architectural improvement attempts, , encrypting the data stored on the cloud computing platform might be the most intuitive one [7]. An encryption algorithm may offer the benefit of minimum reliance on the service provider, but it cannot entirely eradicate privacy leakage to the service provider. During computation, all the encrypted data will be decrypted into plain text in memory that is not protected by the current cloud architecture. Thus, privacy protection inspires research regarding how to query data without revealing information to service provider, like [16, 35, 21, 22]. All of these solutions hide a client's private data from the service provider while the client is retrieving data, but there are problems with this approach, from the point that data retrieved in this manner is not eligible for arbitrary computing. An improved approach allows the service provider to compute with private data on behalf of a client without knowing the content [36]. However, this approach is only applicable for linear programming, and cannot be universally utilized. HP lab also has proposed a mechanism to assist the service provider to conform to privacy law [29]. But this approach is not universally applicable either. It only works perfectly for applications calculating a function of the input that can be expressed as a circuit, and it needs full cooperation from the service provider, which is not guaranteed.

Being treated as the saviour of privacy-preserving data outsourcing, fully homomorphic encryption [12] looks extremely promising by allowing arbitrary computing. But as far as we know, currently there is no homomorphic algorithm that offers acceptable processor overload upon all computations. Although Microsoft has proposed a practical and efficient scheme to perform homomorphic encryption [18], it actually only supports applications that require "somewhat" homomorphic. Namely it's not "full" homomorphic that can be applied universally.

In addition to encrypting data from the service provider, there exist methods enabling privacy auditing as well, such as the privacy-preserving public auditing approach [37]. This approach introduces a trusted third party auditor (TPA) to audit the integrity of outsourced data, and utilizes a homomorphic linear authenticator and random masking to guarantee that the TPA can not gain any information about the data content. The idea of this method sheds a light on the problem brought in by a untrustworthy third party, but is still imperfect facing the mutual-distrust problem in cloud computing – TPA can assure the preservation of privacy for clients, but does not provide monitoring functionality for the service provider.

7. DISCUSSION AND CONCLUSION

When designing a cloud architecture intended for commercial deployment, it is impractical to sacrifice cloud functions for security goals. So any privacy-protected platform should not damage the cloud functions due to any privacy related improvements in the architecture. Because in our architecture we retain all the privileged operations of an ordinary cloud computing node, there is no side effect in cloud functionality. Clients can register, upload and manage their image at will, and the cloud provider can still perform all necessary cloud operations, such as launch, suspend, shutdown and migration. The only difference is that the service provider no longer handles the platform directly. All the requests have to be transferred from the Service Domain to the Neutral Domain, which will then executes the requested service. One desirable feature of this design is that it does not sacrifice any of the original cloud computing functions.

Utilizing the privacy-preserving feature of NeuCloud, we can continue to investigate more powerful domain/VM introspection solutions without being concered about invading a client's privacy. We will conduct our future work upon a domain/VM monitoring mechanism based on the combination of a security-policy database and a privacy-policy database. And in cloud computing environment, sometimes other service functionalities are needed. For example, using policy-based detecting, if the service provider finds out that a client's VM is lacking of a crucial security patch, he can offer a patching service indirectly with the help of the Neutral Domain. And Neutral Domain will always inform cloud clients about the system updates. In this way, the patching process is also more transparent than before.

To conclude, in this paper, we described the motivation, design and implementation of the NeuCloud architecture which enables privacy-preserving monitoring. It solves the mutual distrust problem between a cloud service provider and client. With the help of NeuCloud, a service provider can monitor a client's data and processes to ensure a secure cloud computing environment and avoid any malicious uses of the cloud computing platform, without raising users' concern about the privacy problems. The idea of moving the service provider into a unprivileged Service Domain and setting up the Neutral Domain can be universally implemented on both type-I and type-II virtualization platforms with negligible overhead.

8. ACKNOWLEDGEMENT

This work was partially funded by NSF CNS-1100221.

9. REFERENCES

[1] 0wning xen in vegas! http://theinvisiblethings.blogspot.com/2008/07/0wning-xen-in-vegas.html.

[2] IT cloud services user survey, pt.2: Top benefits and challenges. http://blogs.idc.com/ie/?p=210.

[3] virtioserial. http://fedoraproject.org/wiki/Features/VirtioSerial.

[4] Xenbus. http://wiki.xensource.com/xenwiki/XenBus.

[5] 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, Feb 2009.

[6] S. Berger, R. Cáceres, K. A. Goldman, R. Perez, R. Sailer, and L. Doorn. vtpm: Virtualizing the trusted platform module. In *In USENIX Security*, pages 305–320, 2006.

[7] I.-H. Chuang, S.-H. Li, K.-C. Huang, and Y.-H. Kuo. An effective privacy protection scheme for cloud computing. In *Advanced Communication Technology (ICACT), 2011 13th International Conference on*, pages 260 –265, feb. 2011.

[8] K. Dahbur, B. Mohammad, and A. B. Tarakji. A survey of risks, threats and vulnerabilities in cloud computing. In *Proceedings of the 2011 International Conference on Intelligent Semantic Web-Services and Applications*, ISWSA '11, pages 12:1–12:6, New York, NY, USA, 2011. ACM.

[9] B. Dolan-Gavitt, T. Leek, M. Zhivich, J. Giffin, and W. Lee. Virtuoso: Narrowing the semantic gap in virtual machine introspection. In *Security and Privacy (SP), 2011 IEEE Symposium on*, pages 297 –312, may 2011.

[10] T. Garfinkel, B. Pfaff, J. Chow, M. Rosenblum, and D. Boneh. Terra: a virtual machine-based platform for trusted computing. *SIGOPS Oper. Syst. Rev.*, 37:193–206, October 2003.

[11] R. Gellman. Privacy in the clouds: Risks to privacy and confidentiality from cloud computing. Technical report, World Privacy Forum, 2009.

[12] C. Gentry. Computing arbitrary functions of encrypted data. *Commun. ACM*, 53:97–105, March 2010.

[13] W. Han-zhang and H. Liu-sheng. An improved trusted cloud computing platform model based on daa and privacy ca scheme. In *Computer Application and System Modeling (ICCASM), 2010 International Conference on*, volume 13, pages V13–33 –V13–39, 2010.

[14] B. Hay and K. Nance. Forensics examination of volatile system data using virtual introspection. *SIGOPS Oper. Syst. Rev.*, 42:74–82, April 2008.

[15] X. Jiang, X. Wang, and D. Xu. Stealthy malware detection and monitoring through vmm-based "out-of-the-box" semantic view reconstruction. *ACM Trans. Inf. Syst. Secur.*, 13:12:1–12:28, March 2010.

[16] W.-S. Juang and Y.-Y. Shue. A secure and privacy protection digital goods trading scheme in cloud computing. In *Computer Symposium (ICS), 2010 International*, pages 288 –293, dec. 2010.

[17] E. Keller, J. Szefer, J. Rexford, and R. B. Lee. Nohype: virtualized cloud infrastructure without the virtualization. *SIGARCH Comput. Archit. News*, 38:350–361, June 2010.

[18] K. Lauter, M. Naehrig, and V. Vaikuntanathan. Can homomorphic encryption be practical? Cryptology ePrint Archive, Report 2011/405, 2011. http://eprint.iacr.org/.

[19] C. Li, A. Raghunathan, and N. Jha. Secure virtual machine execution under an untrusted management os. In *Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on*, pages 172 –179, 2010.

[20] D. Li, H. Jin, Y. Shao, and X. Liao. A high-efficient inter-domain data transferring system for virtual machines. In *Proceedings of the 3rd International Conference on Ubiquitous Information Management and Communication*, ICUIMC '09, pages 385–390, New York, NY, USA, 2009. ACM.

[21] J. Li, Q. Wang, C. Wang, N. Cao, K. Ren, and W. Lou. Fuzzy keyword search over encrypted data in cloud computing. In *INFOCOM, 2010 Proceedings IEEE*, pages 1 –5, march 2010.

[22] Q. Liu, G. Wang, and J. Wu. An efficient privacy preserving keyword search scheme in cloud computing. In *Computational Science and Engineering, 2009. CSE '09. International Conference on*, volume 2, pages 715 –720, aug. 2009.

[23] K. E. Mayes, K. Markantonakis, and A. Tomlinson. Introduction to the tpm. In *Smart Cards, Tokens, Security and Applications*, pages 155–172. Springer US, 2008.

[24] J. McHugh, A. Christie, and J. Allen. Defending yourself: the role of intrusion detection systems. *Software, IEEE*, 17(5):42 –51, 2000.

[25] D. G. Murray, G. Milos, and S. Hand. Improving xen security through disaggregation. In *Proceedings of the fourth ACM SIGPLAN/SIGOPS international conference on Virtual execution environments*, VEE '08, pages 151–160, New York, NY, USA, 2008. ACM.

[26] K. Nance, M. Bishop, and B. Hay. Virtual machine introspection: Observation or interference? *IEEE Security and Privacy*, 6:32–37, September 2008.

[27] B. Payne, M. Carbone, M. Sharif, and W. Lee. Lares: An architecture for secure active monitoring using virtualization. In *Security and Privacy, 2008. SP 2008. IEEE Symposium on*, pages 233 –247, may 2008.

[28] B. Payne, M. de Carbone, and W. Lee. Secure and flexible monitoring of virtual machines. In *Computer Security Applications Conference, 2007. ACSAC 2007. Twenty-Third Annual*, pages 385 –397, 2007.

[29] S. Pearson, Y. Shen, and M. Mowbray. A privacy manager for cloud computing. In *Proceedings of the 1st International Conference on Cloud Computing*, CloudCom '09, pages 90–106, Berlin, Heidelberg, 2009. Springer-Verlag.

[30] N. L. Petroni, Jr. and M. Hicks. Automated detection of persistent kernel control-flow attacks. In *Proceedings of the 14th ACM conference on Computer and communications security*, CCS '07, pages 103–115, New York, NY, USA, 2007. ACM.

[31] J. Pfoh, C. Schneider, and C. Eckert. A formal model for virtual machine introspection. In *Proceedings of the 1st ACM workshop on Virtual machine security*, VMSec '09, pages 1–10, New York, NY, USA, 2009. ACM.

[32] M. Price. The paradox of security in virtual environments. *Computer*, 41(11):22 –28, nov. 2008.

[33] T. Ristenpart, E. Tromer, H. Shacham, and S. Savage. Hey, you, get off of my cloud: exploring information leakage in third-party compute clouds. In *Proceedings of the 16th ACM conference on Computer and communications security*, CCS '09, pages 199–212, New York, NY, USA, 2009. ACM.

[34] S. W. Smith and D. Safford. Practical server privacy with secure

coprocessors. *IBM Systems Journal*, 40(3):683 –695, 2001.

[35] C. Wang, N. Cao, J. Li, K. Ren, and W. Lou. Secure ranked keyword search over encrypted cloud data. In *Distributed Computing Systems (ICDCS), 2010 IEEE 30th International Conference on*, pages 253 –262, june 2010.

[36] C. Wang, K. Ren, and J. Wang. Secure and practical outsourcing of linear programming in cloud computing. In *INFOCOM, 2011 Proceedings IEEE*, april 2011.

[37] C. Wang, Q. Wang, K. Ren, and W. Lou. Privacy-preserving public auditing for data storage security in cloud computing. In *INFOCOM, 2010 Proceedings IEEE*, pages 1 –9, 2010.

[38] Z. Wang and X. Jiang. Hypersafe: A lightweight approach to provide lifetime hypervisor control-flow integrity. In *Security and Privacy (SP), 2010 IEEE Symposium on*, pages 380 –395, May 2010.

APPENDIX

A. TPM AND TCB

Specified by the Trusted Computing Group (TCG), the Trusted Platform Module (TPM) chip can be used to authenticate hardware devices [23]. It can be commonly found on almost all the motherboards of servers and high-end PCs. A unique and secret RSA Endorsement Key (EK) is generated for each TPM at the time of manufacture and will be permanently sealed inside the chip, and other sensitive data will be stored into shielded memory. The main role of TPM chips in trusted computing is to act as the Core Root of Trust for Measurement (CRTM), which measures the integrity metrics of modules, holds them in Platform Configuration Registers (PCRs) and reports them in an authenticated way in remote attestation. For privacy concerns, EK is not allowed to be used as platform identity directly. Instead, Application Identity Keys (AIKs) are created to sign these PCR values. TPM is usually used to boot a Trusted Computing Base (TCB). A detailed example to establish TCB with TPM can be found in Terra model [10].

B. PLATFORM SPECIFICATION

The platform specification of our implementation is in Table 3.

Table 3: Implementation platform specifications

Hardware:
Server: IBM x3650 M3
CPU: Intel Xeon x5650 95W 2.66GHz/1333MHz/12MB $\times$ 2
RAM: 8GB DDR3 1333MHz LP RDIMM $\times$ 3
Ethernet: Broadcom NetXtreme II BCM5709 Gigabit Ethernet
TPM version: 1.2.2.60 Switch: IBM 1$\times$8 Console Switch
Software (Type-I virtualization):
Xen version: 4.0.1-21326-02-0.5
Control VM/Service VM OS: openSUSE 11.3 x86_64
Control VM/Service VM kernel: 2.6.34.7-0.7-xen
Software (Type-II virtualization):
Host OS: openSUSE 11.4 x86_64
Host OS kernel: 2.6.37.1-1.2-desktop
Service VM OS: openSUSE 11.4 x86_64
Service VM kernel: 2.6.37.1-1.2-desktop