2012

TOWARDS AN INCENTIVE COMPATIBLE FRAMEWORK OF SECURE CLOUD COMPUTING

Yulong Zhang
Virginia Commonwealth University

TOWARDS AN INCENTIVE COMPATIBLE FRAMEWORK OF SECURE CLOUD COMPUTING

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science at Virginia Commonwealth University.

by

Yulong Zhang

Master of Science
in
Computer Science

Director: Meng Yu
Associate Professor, Computer Science Department

Virginia Commonwealth University
Richmond, Virginia
April 2012

ACKNOWLEDGEMENT

This work would not be possible without the support of my family, my friends, my advisor, and all others who helped me during the writing of this thesis.

First and foremost, I would like to show my deepest gratitude to my advisor, Dr. Meng Yu, who helps me to plan and build an academic career. He always offers guidance to me about how to suite into the academic circles, and is willing to provide me opportunities to attend conferences with the fee covered by his fundings. Aside from the research guidance, he also gives me chances to take part in paper review processes of conferences. He is the one leads me into this fantastic world of security and privacy research. I also would like to show my great thanks to Dr. Wanyu Zang, who helped me a lot with both my research and course works. Dr. Meng Yu and Dr. Wanyu Zang are like my best friends in daily life. They provided me with very useful suggestions and saved me from losing myself when I was lonely, frustrated, upset and in despair. I really appreciate that.

Here is a special thanks to Chao Ni, who didn't give up believing in me when I came down in the world. All along, you have been my soul mate, and known me so well. I can't see clearly the road ahead, but you will always be one of the reasons influencing my life decisions, just like the day I quitted my M.S. degree in Peking University and came here for our dreams. C'est la vie avec toi.

My lab mates, my courses instructors, and my thesis committee members: thanks so much for sparing your precious time offering me so many important advices. I will never forget them.

And finally, thanks, my mother and father. Without you, without me.

Contents

1 Introduction **1**

1.1 Cloud Computing 1

1.2 Incentive Compatible Problems 2

2 NeuCloud: The Trusted Computing Base for Incentive-compatible Privacy Trading **6**

2.1 Design Goals 9

2.2 Background 10

2.2.1 Root of Trust Measurement 10

2.2.2 System Management Mode 12

2.3 NeuCloud Overview: An In-memory Private Space For Transparent Privacy Trading 13

2.3.1 Trusted Boot Procedure 16

2.3.2 Secure and Private Interaction 18

2.4 Formalization of Privacy Trading 20

2.5 Implementation and Evaluation 25

2.6 A Variation of NeuCloud 26

2.7 Summary 30

3 No free-riders: Incentive Compatible Moving Target Defense Against VM-Colocation Attacks in Clouds 31

3.1 Preliminaries 34

3.1.1 Assumptions and Notations 34

3.1.2 Incentive Compatible Game 35

3.2 VM-migration Based Moving Target Defence 36

3.2.1 The Optimal Number of Moving VMs 36

3.2.2 The Constant Cost of Migration 41

3.2.3 Defense Timeline 43

3.3 Evaluation 43

3.3.1 Security 43

3.3.2 Practicability 45

3.4 Summary 45

4 Conclusion and Future Works 47

List of Figures

1.1 A general cloud computing architecture. 2

1.2 The mutual relationships in a typical public cloud. 3

2.1 The philosophy of NeuCloud: partially and transparently disclosure of privacy. . . . 8

2.2 The schematic view of RTM boot process. 11

2.3 The system mode transition graph (using Intel as an example) 11

2.4 Default SMRAM Memory Map 12

2.5 The basic architecture of NeuCloud. 13

2.6 The functional modules of NeuCloud. 14

2.7 The trusted boot procedure of NeuCloud. 16

2.8 The runtime schematic view. 18

2.9 The illustration of NeuCloud's information flow 21

2.10 The relationship between the execution time and the interval to call NeuCloud ($N = 10000, \zeta = 50\%$). 26

2.11 The relationship between the execution time and the percentage of tasks executed in NeuCloud ($N = 10000$). 27

2.12 A variation of NeuCloud: the architecture. 28

2.13 A variation of NeuCloud: the implementation 29

3.1 Shamir's secret sharing in cloud environment and the moving target defense through VM-migration. 32

3.2 The k-N relationship when n=25, $\bar{c} = 4$, and m=γ varies from 10 to 7. 37

3.3 The influence of migration to VM computation and web service. 42

3.4 The timeline of attacks and moving target defense (VM migration). 44

3.5 The influence to k_{opt} and corresponding N_{max} by adjusting $\bar{c}$ or γ in Equation 3.5. . . 46

List of Tables

2.1 Specifications of execution time measurement. 25

3.1 Platform specifications of migration downtime experiment. 41

3.2 Amazon EC2 Linux/UNIX instances pricing and specifications (US East, up to January 2012) [2] 45

ABSTRACT

TOWARDS AN INCENTIVE COMPATIBLE FRAMEWORK OF SECURE CLOUD COMPUTING

By Yulong Zhang, M.S.

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science at Virginia Commonwealth University.

Virginia Commonwealth University, 2012.

Major Director: Meng Yu, Ph.D.

Associate Professor, Virginia Commonwealth University

Cloud computing has changed how services are provided and supported through the computing infrastructure. It has the advantages such as flexibility , scalability , compatibility and availability . However, the current architecture design also brings in some troublesome problems, like the balance of cooperation benefits and privacy concerns between the cloud provider and the cloud users, and the balance of cooperation benefits and free-rider concerns between different cloud users. Theses two problems together form the incentive problem in cloud environment.

The first conflict lies between the reliance of services and the concerns of secrets of cloud users. To solve it, we proposes a novel architecture, NeuCloud, to enable partially, trusted, transparently, accountably privacy manipulation and revelation. With the help of this architecture, the privacy-sensitive users can be more confident to move to public clouds. A trusted computing base is not enough, in order to stimulate incentive-compatible privacy trading, we present a theoretical framework and provide the guidelines for cloud provider to compensate the cloud user's privacy-risk-aversion. We implement the NeuCloud and evaluate it. Moreover, a improved model of NeuCloud is discussed.

The second part of this thesis strives to solve the free-rider problem in cloud environment. For example, the VM-colocation attacks have become serious threats to cloud environment. We propose to construct an incentive-compatible moving-target-defense by periodically migrating VMs, making it much harder for adversaries to locate the target VMs. We developed theories about whether the migration of VMs is worthy and how the optimal migration interval can be determined. To the best of our knowledge, our work is the first effort to develop a formal and quantified model to guide the migration strategy of clouds to improve security. Our analysis shows that our placement based defense can significantly improve the security level of the cloud with acceptable costs.

In summary, the main objective of this study is to provide an incentive-compatible to eliminate the cloud user's privacy or cooperative concerns. The proposed methodology can directly be applied in commercial cloud and help this new computing fashion go further in the history. The theoretical part of this work can be extended to other fields where privacy and free-rider concerns exist.

Chapter 1

Introduction

1.1 Cloud Computing

Cloud computing [7] is becoming a major trend in computing services with its inspiring features of elastic "data anywhere" and "computing anywhere". Generally, there are three types of cloud computing services: Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS) and Software-as-a-Service (SaaS). Among them the IaaS is the most fundamental one, where a cloud user owns a virtual machine (VM) and purchases virtual power to execute as needed, just like running a virtual server. A typical example of public IaaS is the Amazon Elastic Computing (EC2) Services [2], of which the abstraction of architecture is shown in Figrue 1.1.

The architecture shown in Figure 1.1 is also used by Eucalyptus [3], OpenNebular [5], and many other open-source cloud products [12]. In the architecture, a Cloud Controller (CLC) provides cloud interfaces to clients. The Cluster Controller (CC) manages the intranet and schedules Virtual Machines (VMs) execution on nodes belonging to it, and each Node Controller (NC) directly controls VMs instances that it hosts. On a node with Type-I virtualization, a Virtual Machine Monitor (VMM), such

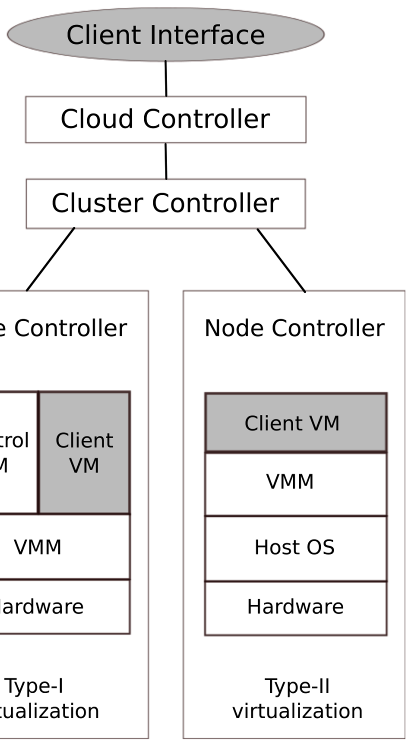


Figure 1.1: A general cloud computing architecture.

as Xen [10], abstracts the hardware resources and replace all the direct communications between VMs and hardware to hypercalls, endowing management authority to a special control VM, e.g., Dom 0. On a node with Type-II virtualization, a host operating system adds a module functioning as VMM, such as KVM [20], to manage guest VMs. In the current architecture, there is no separation of privilege design to deprive unnecessary privileges from the service provider. The service provider owns the hardware, hypervisor, and control VM, becoming the *Dictator* of the cloud.

1.2 Incentive Compatible Problems

This IaaS architecture has its advantages such as flexibility (VMs can be easily migrated to every corner of the cloud, without shutting down the service), scalability (a huge population of VMs can be integrated into the cloud and some of them can group into clusters), compatibility (VMs can be

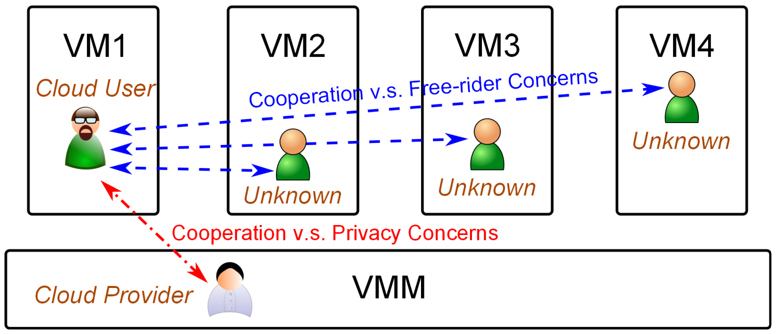

VM1
Cloud User
VM2
Unknown
VM3
Unknown
VM4
Unknown
Cooperation v.s. Free-rider Concerns
Cooperation v.s. Privacy Concerns
Cloud Provider
VMM

Figure 1.2: The mutual relationships in a typical public cloud.

launched in the virtualization environment regardless of the beneath hardware and the instruction set as long as their Operating Systems are supported by the VMM) and usability (many services can be outsourced to VMM, e.g. Intrusion Detection Service (IDS) [35], to minimize cost and management efforts). However, this architecture design also brings in some troublesome problems, like the balance of cooperation benefits and privacy concerns between the cloud provider and the cloud users, and the balance of cooperation benefits and free-rider concerns between different cloud users. Theses two problems together form the incentive problem in cloud environment.

The Balance of Services and Secrets

As illustrated in Figure 1.2, from a cloud user's view, typically there are three roles inside the IaaS public cloud: the cloud user himself/herself, the other cloud neighbour VMs owned by unknown parties, and the cloud provider.

Taking the advantages of cloud computing, the cloud user can outsource part of his/her computation or data to the cloud provider. Unavoidably, the secrets inside the user's data might be seen by the cloud provider. We admit that those commercial cloud providers would sign a service contract with cloud users, which clearly indicates the responsibility and liability of cloud provider regarding to the privacy protection. However, the policies written on paper cannot make the users more confident: with cloud provider alone as the "dictator" in the cloud environment, who can supervise the operation of the cloud provider? How to trace the manipulation of their privacy given that cloud users have no control at all towards their data in cloud? Even if the cloud provider is well-meant, how to prevent its employee to launch insider-attacks? Up to now these answers still remain agnostic to commercial cloud services.

Then, someone might ask, what if the cloud user chooses not to share any secrets with the cloud provider and restricts his/her computation/data in an isolated or encrypted domain? This suggestion seems promising at first glance. However, why do the cloud users bother to use cloud if they don't make use of it? They can instead use the local resources to achieve the same goal, which saves the communication cost and the isolation/encryption cost. As a matter of fact, following this philosophy, some bank and government organizations turn to the private cloud solutions instead of the above public IaaS approach. Nevertheless, for small users, whose purpose is to outsource daily computing, public cloud is still the most economical alternative and thus the service-privacy conflict still remains a serious task as well as challenge for researchers.

The Balance of Contribution and Gain In Cooperation

Except the service-privacy trade-off between cloud users and cloud provider, another issue shown in Figrue 1.2 is the cooperation game between cloud users themselves. On one hand, Each cloud user has no clue about his/her neighbours; they might be benign but they might be malicious either. In order to defend such kind of internal attacks, the cooperation of different cloud users becomes necessary (we will discuss about a certain example in Chapter 3). At the same time, there are enormous number of attacks launched outside the cloud, it must be more effective if all the cloud users can unite with each other to defend those attacks. As a whole, the cooperation of cloud users should improve the overall utility. However, on the other hand, none of the rational individuals would prefer to contribute; cloud users expect the benefit of cooperation but none of them would like to pay for the public utility shared by the overall community. In the traditional cloud computing envrionment, everyone has the incentive to be the free-rider. Thus it becomes the second serious challenge.

The above two problems are tightly connected with each other. The conflict between outsourcing service and the concerns of privacy disclosure reflects the subtle one-to-one relationship between the cloud provider and each cloud user, which is the root incentive-compatible problem in cloud environment. Only if the root trust problem is solved, will the cloud provider be able to act like the arbiter to further mitigate the conflict between cloud users. So on one hand, the two problems describes two parallel conflicting relationships; on the other hand, the solution of the latter relies on the settlement of the former. The goal of this thesis is to address the above two challenges one by one. The service-privacy conflict is analysed in Chapter 2, where a novel cloud computing architecture and a theoretic framework are proposed to provide trusted and transparent privacy exchange. With the trusted cloud provider given in Chapter 2 as the arbiter, a game theory based solution of the free-rider problem is discussed later in Chapter 3. As a specific case, the VM moving-target defence is analysed and incentive-compatible cooperative defence framework is developed for it.

Chapter 2

NeuCloud: The Trusted Computing Base for Incentive-compatible Privacy Trading

Cloud computing [7] 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 [4]. 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 [18]. Therefore, many new security mechanisms have been developed to protect users' privacy. Many techniques have been proposed to protect data privacy through cryptography or system-level modifications [13, 23, 46, 39, 44, 17, 28, 34]. The unified goal of the above approaches is shown in the left part in Figure 2.1, which aims to hide 100% of users' content from the cloud provider.

To name a few, some methods strive to thoroughly eliminate Hypervisors and rely on the tamper-resistant processor to provide virtualization, as introduced by "NoHype" [25]. This kind of approach is probably the ultimate scheme to defend malicious or compromised service provider. Apart from the architectural improvement attempts, , encrypting the data stored on the cloud computing platform is also promising [13, 23, 45, 29, 31]. All of these solutions hide a client's private data from the service provider while the client is retrieving data. An improved approach is the fully homomorphic encryption [19], which allows computation on encrypted data. Following the steps, Microsoft has proposed a practical and efficient scheme to perform homomorphic encryption [27].

However, a full encrypted or isolated view from the cloud provider is unacceptable; many cloud services should be carried out with the sacrifice of privacy, due to the following compelling reasons:

Cloud Security Considerations

The cloud monitoring is necessary to protect the cloud environment from the VM-to-host attacks and the cross-VM attacks. 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 [42]. 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) [33, 35, 38] 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 [15]. 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 [40]. 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

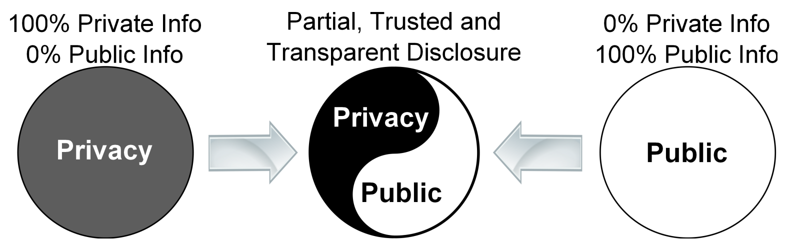


Figure 2.1: The philosophy of NeuCloud: partially and transparently disclosure of privacy.

the privacy requirement of a client. But when we reconsider about the problem, 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: **Is it necessary to monitor all the data and processes of the VMs?** Not all of cloud users' data and programs are privileged enough to arouse serious attacks. Since he full-monitoring scheme has dispelled so many privacy-concerning users, cloud provider should re-consider the trade-off of doing so. From this perspective, cloud provider may balance the degree of monitoring to attract more users.

Performance and Economics Considerations

Some system-level approaches, like the "NoHype" [25], requires hardware modifications and that each VM should occupy a unique core, of which the cost is still unacceptable to most service providers nowadays running on common commercial CPUs. The encryption based protection may be relatively cheap, but it either restricts the computing functions (normal encryption methods are not eligible for arbitrary computing) or the performance (in the case of the homogeneous encryption methods). Thus a 100% protection of private information is somehow impractical.

As a whole, either the totally public scheme or the totally private scheme is acceptable. We need a hybrid solution as shown in Figure 2.1. The philosophy is not only about "half public and half private" but also about "trading privacy" at any rate, so that the user can selectively and confidently exchange their privacy with the cloud provider. All the requirements of this philosophy will be described in Section 2.1.

2.1 Design Goals

The NeuCloud design should satisfy the following goals:

1. **Being able to protect users' privacy from the cloud provider.** This is the most significant requirement. Only if secrets can be effectively hidden from the view of the privileged "dictator", should the privacy-sensitive users turn back to the public cloud. Nevertheless, this is also the most challenging target. It can never be easy to hide secrets from a dictator right in his/her kingdom.

2. **With the ability to isolate VMs.** Protecting privacy from the cloud provider is not the only assurance to be achieved. NeuCloud should also be able to isolate information from different VMs, unless they explicitly want to share it with each other.

3. **Without losing common functions, including security monitoring.** It is good to have a full protection upon privacy. But as we have discussed above, a hybrid scheme should be provided so that only the crucial secrets are protected and the other functional and security-related entities should be able to be exposed.

4. **Enabling transparent and accountable operation on privacy.** Protection is not enough; we should convince the users how well they are protected and who should be blamed if disclosure

happens. NeuCloud should be able to transparently and fairly reveal such information and log all the operations upon secrets.

5. **Making maximum use of legacy platforms.** Many talent ideas have been proposed by the academics by never been applied ever since. The reason is that they are too expensive for the industry. NeuCloud should take advantages from legacy platforms as much as possible.

6. **Secure enough to be the TCB.** The last but not the least, as a Trusted Computing Base (TCB), NeuCloud must be secure enough. That is to say, it has to protect itself from being compromised.

2.2 Background

2.2.1 Root of Trust Measurement

In order to protect the Trusted Computing Base (TCB) from being compromised, the Root of Trust Measurement (RTM) mechanism is used (Figure 2.2), which mainly relied on the Trusted Platform Module (TPM) chip. Specified by the Trusted Computing Group (TCG), the TPM chip can be used to authenticate hardware devices [32]. 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 Privacy CA (Certificate Agency) can authenticate a TPM according to its public Endorsement Key. 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. A detailed example to establish TCB with TPM can be found in Terra model [17].

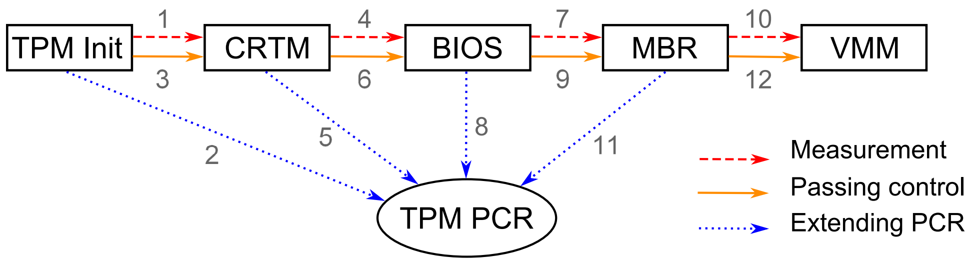


Figure 2.2: The schematic view of RTM boot process.

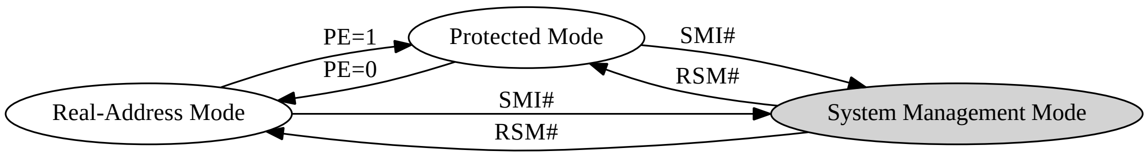


Figure 2.3: The system mode transition graph (using Intel as an example)

Figure 2.2 shows the boot sequence based on RTM: after system boot or reset, the TPM chip gets initialized. It measures the CRTM block in BIOS and extends the TPM PCR; after that the CRTM code is executed and then the BIOS, the MBR, and finally the OS (VMM for cloud platform). During each transition, the measurement value of the next step is always extended into PCR by the following operation:

$$PCR := SHA - 1(PCR + measurement)$$

A new measurement value is concatenated with the current PCR value and then hashed by SHA-1. The result will be stored as a new value of the PCR.

For normal platforms, user can access TPM chip via common driver interfaces. However, in the virtualization environment, user cannot directly talk to TPM chip from a VM. Under such circumstance, vTPM [11] is carried out to realize RTM support.

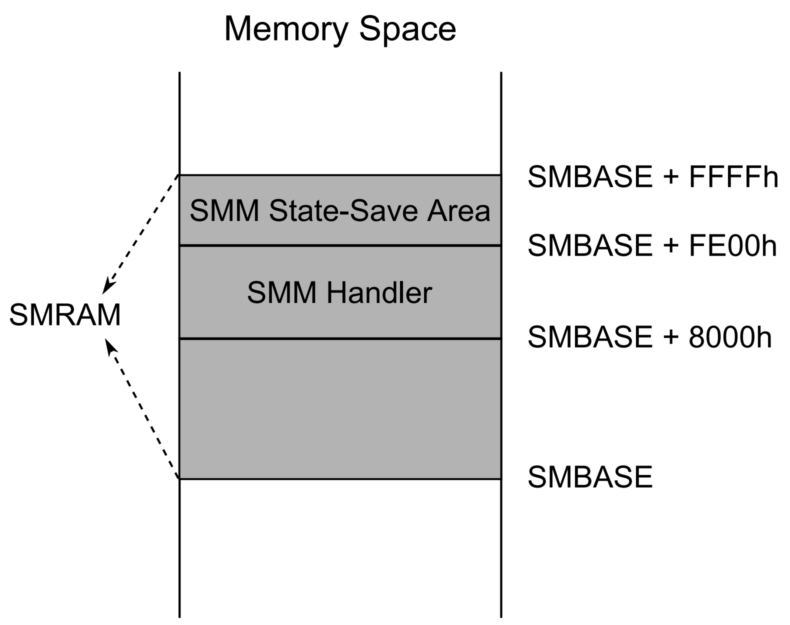


Figure 2.4: Default SMRAM Memory Map

2.2.2 System Management Mode

Both Intel and AMD CPUs support the (System Management Mode)SMM as one of its operating modes. The processor enters the SMM when receiving an SMI, as shown in Figure 2.3. Upon an SMI, the processor saves its state to a dedicated state save map and switches to the SMM. To return from the SMM, the special instruction RSM restores the saved processor state and resumes normal execution.

SMM code is stored in a designated memory called SMRAM 2.4. To provide protection of the SMM code and data, both AMD and Intel provide the capability of locking the SMRAM. When the SMRAM is locked, all accesses to it, except from within the SMM, are prevented. All interrupts, including non-maskable ones, are disabled upon entering the SMM. Thus, no other code running on the system can interfere with the SMI handler. Current hardware can support up to 4 GB of SMRAM. The application of SMM in cloud security has gradually become popular, and it is well accepted by the academia that SMM is an excellent stealthy secure environment [22, 8, 9].

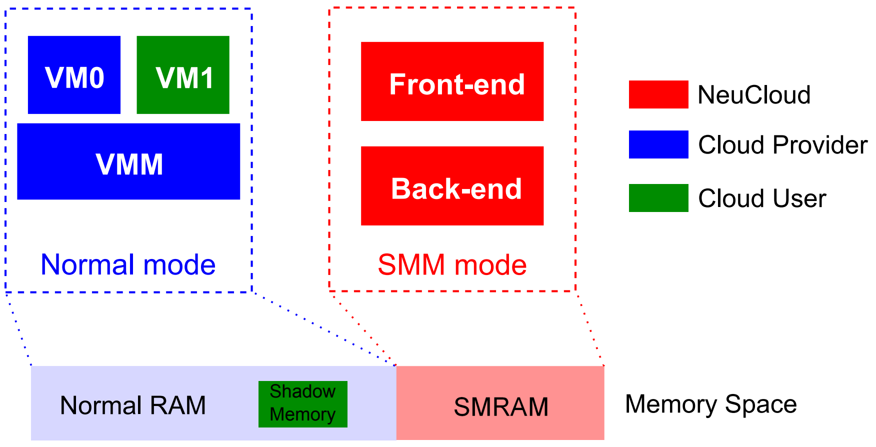


Figure 2.5: The basic architecture of NeuCloud.

2.3 NeuCloud Overview: An In-memory Private Space For Transparent Privacy Trading

Figure 2.5 provides an overview of the NeuCloud architecture. We use the "Type-I" virtualization architecture as an example, and the other type of virtualization platforms can be modified in the similar fashion. The SMRAM is protected from the view of either the cloud provider or the cloud users, and is only visible to the special SMM handlers, which will be introduced later in this section. The rest of the memory space, denoted as "normal RAM" here, is managed by a legacy VMM with NeuCloud driver installed. The NeuCloud driver is unnecessary to be included into TCB as regards privacy protection, but is required to get full NeuCloud support. The privileged "VM0" ("Domain0" in Xen architecture) is kept unmodified, so are the unprivileged cloud users' VMs, except that a NeuCloud client should be installed in the users' VMs if they need NeuCloud functions.

The "VM1" in Figure 2.5 represents one of the cloud users' VMs. Since the VM's memory space is virtualized via the shadow paging mechanism [21], theoretically the private information inside "VM1" is not a secret to VMM at all. This is the exact root cause keeping those privacy-sensitive users away from the public clouds, where their secrets are transparent to many potential adversaries, including the

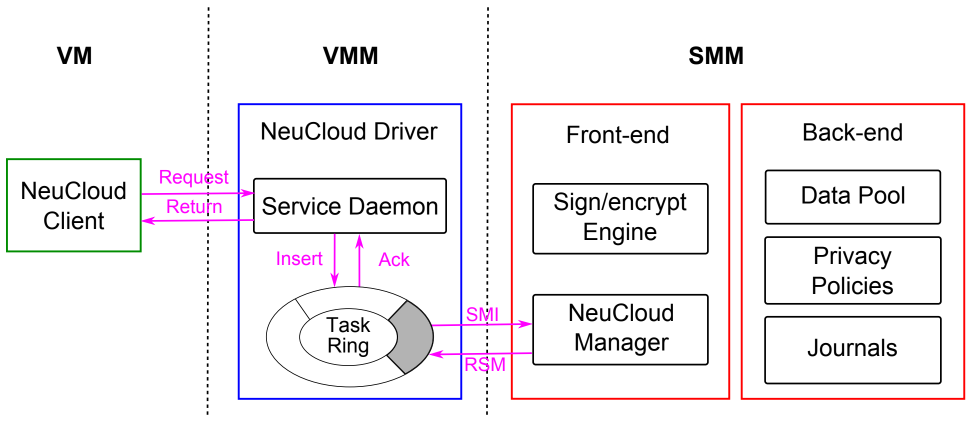


Figure 2.6: The functional modules of NeuCloud.

cloud provider, the attackers being able to compromise the VMM, and the neighbour VMs who can eavesdrop on them [42]. To get avoid of this problem, a logical “private cloud” is inserted beneath the public cloud: the NeuCloud. By “private cloud”, we don’t mean the NeuCloud layer privately belongs to some parties; it is a neutral domain impartially and transparently managing the privacy and the corresponding polices.

To realize NeuCloud, we program the BIOS and add some special SMM handlers, including the front-end modules and the back-end modules, as shown in Figure 2.6. As we mentioned, the absence of NeuCloud client (in users’ VMs) and the NeuCloud driver (in VMM) will not influence the public cloud functions; but in order to enjoy the private cloud services, they are required. The NeuCloud driver adds a new service daemon into the VMM, which is a new hypercall for Xen in particular. Users can call the service daemon via the NeuCloud client, and the NeuCloud driver will continue to deliver the data and corresponding operations to the NeuCloud layer. Only a fixed set of operations are accepted to avoid unknown malicious exploitations. Since the NeuCloud manager will further parse and authenticate the requested operations, the compromise of the NeuCloud driver will not lead to privacy leakage or data damage. All the delivered data is encrypted from the cloud provider’s view, which will be discussed later in this section. Upon receiving the service requests, the service daemon will update the current task pool. For simplicity, we design a ring structure to fairly schedule the tasks based on the roundrobin algorithm. Each VM owns a one-operation slice in the ring, and new operations requested by a VM will be inserted into the task FIFO queue corresponding to its unique time slice [1]; 3) when the task pointer active one task, if the operation doesn't carry any new data, the NeuCloud driver will invoke SMI entering into SMM and perform the operation (copy, delete, encrypt, decrypt, add-policy, delete-policy, etc.); 4) if the operation does carry new data (not existing inside SMRAM), data should be first of all prepared as encrypted blocks and the operation parameters should contain the block size. This will divide an operation into several tasks, and for each task the NeuCloud manager only takes data with sizes under a predefined maximum value. The reason of processing limited size of data each time is that SMM diables timer interrupt so that timing is impossible; for the same reason the pre-emption from SMM is also impossible. We use the limited-time-processing trick to circumvent this drawback of SMM, and it won't bring in any influence to cloud users since most of the popular cryptography methods are based on block encryption/decryption.

Once received the tasks, the NeuCloud manager will parse and authenticate them. Two categories of tasks are allowed: mutual authentication, which includes task originator and finisher authentication, and new users' key exchange for future task authentications; data and policies manipulation, which includes encryption/decryption, data and policies modification, policies based privacy sharing, etc. All the operations will be logged and written into corresponding journals, thus accountability is guaranteed.

The overall NeuCloud design is a complex and challenging.In the following discussions we will describe the internal mechanisms in detail. The notation used in this section is as follows:

1. $message$: Any objects that needs to be encrypted;
2. $\{message_1, message_2\}$: Concatenation of messages;
3. $(message)_{SK_{owner}}$: Encryption with owner's symmetric/session key;

[1]An operation is the basic execution unit for cloud users. The NeuCloud daemon will further divide an operation into several tasks depending on the data size.

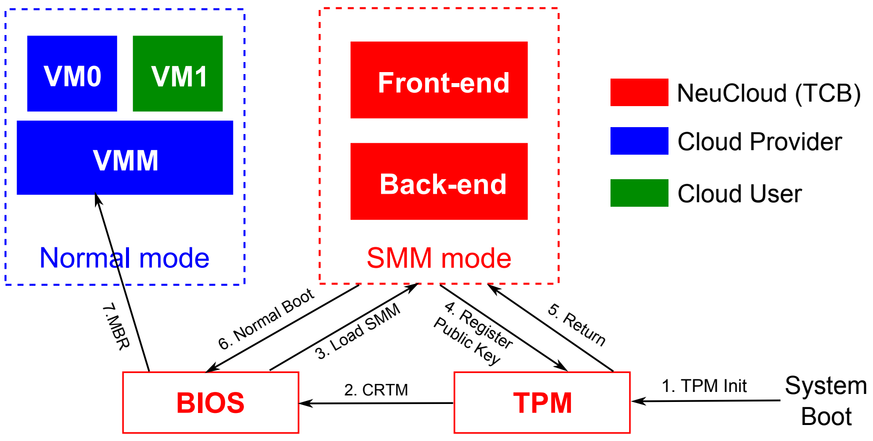


Figure 2.7: The trusted boot procedure of NeuCloud.

4. $(message)_{K^{Pub}_{owner}}$: Encryption with owner's public key;

5. $[message]_{K^{Pri}_{owner}}$: Signing with owner's private key;

6. $Cert(K^{Pub}_{owner})$: The certificate of owner's public key.

2.3.1 Trusted Boot Procedure

The first challenge is how to boot the NeuCloud along with the legacy virtualization system. As shown in Figure 2.7, upon system boot or reset, the TPM chip is initialized. It then measures the Core Root of Trust for Measurement (CRTM) in BIOS boot block, which will further measures the SMM handlers that will be written into SMRAM. Note that all the measurement results will be extended into the Platform Configuration Register (PCR). Except the common boot operations, the BIOS boot block needs to setup the SMRAM region and copy SMM handlers (front-end modules and back-end modules)into the SMRAM. After that the BIOS hands over the platform control to the boot code in SMM by calling SMI. The SMM boot code will then generate a fresh asymmetric key-pair (K^{Pub}_{SMM}, K^{Pri}_{SMM}), and extend the public key information into the PCR. Later all the cloud users can easily obtain this public key from TPM. Finishing the initialization of SMM handlers, NeuCloud will pass the control back to BIOS and continue the normal boot procedure through the Master Boot Record (MBR), loading the VMM and other legacy components. Afterwards, all the users on the platform are able to request for attestation at any time. Cloud provider ("VM0" and VMM) can simply interact with the TPM chip, while cloud users have to go through the vTPM [11] channels. The attestation procedure is:

1. On receipt of a request for attestation challenge, the TPM chp will respond with the public Attestation Identity Key (AIK^{Pub}) and the certified public Endorsement Key (EK^{Pub}). The challengers can further validate the certification of the EK^{Pub} by forwarding the keys to the Privacy CA [32].

   $NeuCloud \rightarrow Challenger \rightarrow PrivacyCA : \{AIK^{Pub}, EK^{Pub}\}$

2. If the certification of the EK^{Pub} is valid, the Privacy CA signs a certificate for AIK^{Pub}, and encrypts the certificate with a newly created session key SK_{PCA}. Along with them, SK_{PCA} and AIK^{Pub}, encrypted by EK^{Pub}, are altogether sent back to the challenger. The challenger then deliver the second blob to the TPM chip.

   $PrivacyCA \rightarrow Challenger : \{([Cert(AIK^{Pub})]_{K^{Pri}_{PCA}})_{SK_{PCA}}, (\{SK_{PCA}, AIK^{Pub}\})_{EK^{Pub}}\}$

   $Challenger \rightarrow NeuCloud : (\{SK_{PCA}, AIK^{Pub}\})_{EK^{Pub}}$

3. Once received them, the NeuCloud TPM decrypts the SK_{PCA} and AIK^{Pub} using its $EK^{(}Pri)$, and checks if the AIK^{Pub} matches with the one it owns. If everything goes well, the NeuCloud releases the session key SK_{PCA}.

   $NeuCloud \rightarrow Challenger : SK_{PCA}$

4. With the session key, the challenger can obtain the signed certificate of AIK^{Pub}. Now that AIK^{Pub} is authenticated, the user can generate a random number n to perform the real-time platform attestation by asking for the signed PCR value along with this one-time random number.

   $Challenger \rightarrow NeuCloud : (n)_{EK^{Pub}}$

5. The TPM chip should reply with the Stored Measurement Log (SML, containing the BIOS, MBR

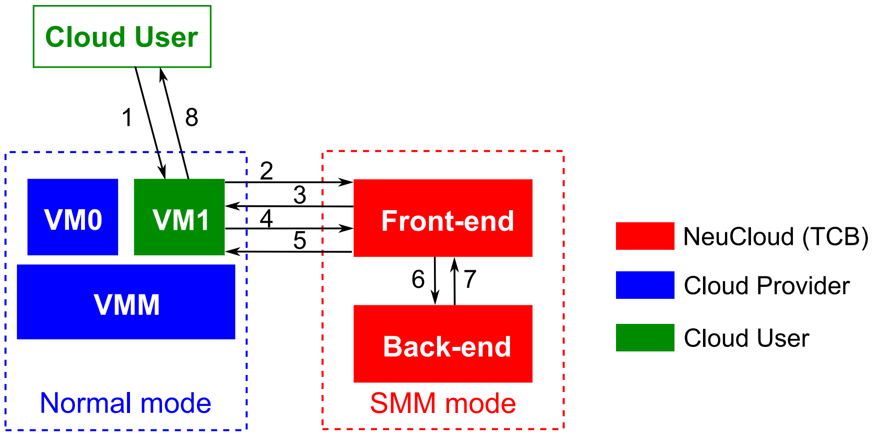


Figure 2.8: The runtime schematic view.

and VMM measurement fingerprints as well as the public key of NeuCloud SMM handlers, K_{SMM}^{Pub}), along with the PCR value and the random number signed by AIK^{Pub}.

$NeuCloud \rightarrow Challenger : \{[\{PCR, n\}]_{AIK^{Pub}}, SML\}$

6. Upon receiving them, the challenger first of all check if n is correct, and then applying the PCR's extend operation on SML (to see if it is the same with the received PCR value). If everything matches, and the platform measurement fingerprints in SML satisfying the challenger's requirement, the attestation is finished with success and the K_{SMM}^{Pub} can be confidently used in the later authentication.

2.3.2 Secure and Private Interaction

Another challenge is how to verify the task originator and finisher. Different users may interact with NeuCloud, then how to know which key should be used for the current task? Moreover, since all the VM's operations should bypass the VMM, how can the users make sure if the "NeuCloud" they are interacting with is actually disguised by the VMM? It is possible that the cloud provider plays as a middle-man to intercept secrets. To deal with the above problems, we need a secure and private interaction protocol, illustrated in Figure 2.8.

1. The cloud user securely communicate with VM1, for example by SSH. The data in VM1 are classified into two categories: public, and private, of which the latter is pre-encrypted by the symmetric key SK_{user} before being uploaded to the cloud. The SK_{user}, as well as the user's public key K^{Pub}_{user}, should also be agnostic to the cloud provider so it should be stored in VM1 encrypted with K^{Pub}_{SMM}.

2. If it is the first time for VM1 to register itself to NeuCloud, it should request a registration operation with corresponding keys and a random operation identity number m. VMM will add this operation into task ring along with a unique ID of VM1. It doesn't matter what the value of ID is as long as it is consistent during VM1's lifetime.

   $VM1 \rightarrow VMM \rightarrow NeuCloud : \{ID, (\{m, SK_{user}, K^{Pub}_{user}\})_{K^{Pub}_{SMM}}\}$

3. The NeuCloud manager parses the task as registration and ask the sign/encrypt engine to decrypts the keys of VM1. After that the manager stores the SK_{user} and the K^{Pub}_{user} in the ID entry. Now that this operation has been finished, the manager returns to the outer world with signed operation identity and the status marker ACK.

   $NeuCloud \rightarrow VMM \rightarrow VM1 : \{ACK, [\{ID, m\}]_{K^{Pri}_{SMM}}\}$

4. After registration the VM1 can freely launch any data or policies related operations. Large operations will be divided into small tasks by NeuCloud daemon in VMM.

5. A task's accomplishment will pop out the task from the task queue. The task ring circles to other VMs until it turns back to VM1. Then the next task of VM1 will be executed.

6. If a task is data manipulation or privacy policies updating, the front-end modules of NeuCloud will send the workload to the back-end modules. All the workload in back-end modules will be logged as a journal.

7. The transitions between VMM and NeuCloud will keep going until the whole operation requested by VM1 is finished. Then the NeuCloud replies with the *ACK*, telling VMM to inform the VM1 of the operation accomplishment.

8. Whether data flowing into NeuCloud should be encrypted or decrypted depends on the users' requests. However, if the users fetch data out of the NeuCloud, the encryption of secrets claimed by the privacy policies is enforced, where SK_{user} is used. The information flow of NeuCloud is depicted in Figure 2.9. User can split their data into public data and private data, with private data encrypted before uploading to the cloud. Private data can be copied into NeuCloud and, optionally, be decrypted later. To fetch private data out, NeuCloud will always perform the encryption process first. We note that it is not the encrypted status or the decrypted status that determines the privacy labels of data; it is possible that the user decrypts his private data in SMRAM just for performing operations on them. As illustrated by Figure 2.9, a decrypted data can either be mapped as public objects("Decrypted Private Data 1"), or be mapped as private objects("Decrypted Private Data 2").

   With NeuCloud, the user is confident about trading privacy with the cloud provider, or other tenant service providers on the same platform: because the execution of data/policies manipulation is authenticate-able, the operations are logged and thus accountable, and the privacy is well protected from all the cloud computing entities including the most privileged one, the cloud provider. Based on NeuCloud, we can further design the incentive privacy trading mechanism for cloud computing.

2.4 Formalization of Privacy Trading

Now that we have an excellent TCB to provide transparent privacy manipulations, we can continue to investigate into the users' reaction towards privacy revelation. The first task is to formalize the privacy

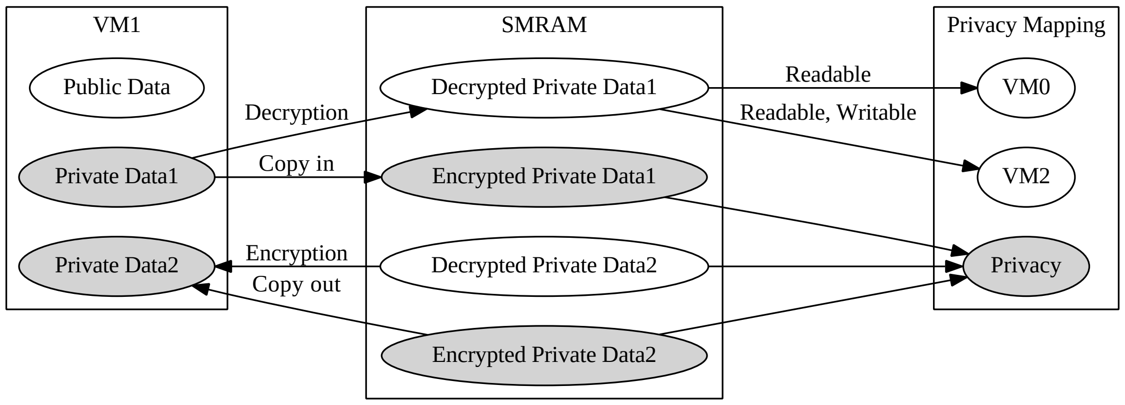


Figure 2.9: The illustration of NeuCloud's information flow .

sensitivity, namely the willingness to reveal the privacy. Consider the following situation: a cloud user (U) rents computing resources from a cloud provider (P) and shares his private information to P in order to achieve some target functions (e.g. data analysis). Of course in NeuCloud this user U has the alternative choice to protect his privacy without the cloud provider's participation.

U can never know how trusty the cloud provider P is. Although U knows how much privacy is released, he has no idea of how the private information is used: it is absolutely possible that P maliciously makes profit from selling the privacy to the adversaries of U. In order to quantify the cost of privacy revelation, we define the revelation set of privacy and the the sensitivity of privacy as follows:

Definition 1 *(Privacy revelation set) The set of private (or credential) items of U that can be requested and released is denoted as* $\Omega = \{\omega_1, \omega_2, ..., \omega_\pi\}$*, where* π *is the total number of the private items. The privacy revelation set* $\wp(\Omega) = \{\rho_0, \rho_1, ..., \rho_\gamma\}$ *is defined as the powerset of* Ω*, which consists of all the possible revelation combinations of the private items. Each* $\rho_i \in \wp(\Omega)(i = 0, 1, ..., \gamma)$ *is called a privacy revelation portfolio, and* γ *is the cardinality of* $\wp(\Omega)$*. Specially,* $\rho_0 = \emptyset$ *and* $\rho_\gamma = \Omega$*.*

Cloud users may have different valuations on different privacy revelation portfolios. The valuation of each portfolio is measured by the monetary mapping $V(\rho_i), i = 0, 1, ..., \gamma$. For any $\rho_i \in \wp(\Omega), i = 0, 1, ..., \gamma$, we have $V(\rho_i) \geq 0$ and specially $V(\rho_0) = 0$. Suppose $0 \leq i \leq j \leq \gamma$, $V(\rho_i) > V(\rho_j)$

is equivalent to the proposition "ρ_i is valued higher than ρ_j". Similarly, $V(\rho_i) < V(\rho_j)$ means "ρ_i is less valued than ρ_j", and $V(\rho_i) = V(\rho_j)$ reveals the equal valuation of U upon ρ_i and ρ_j. Generally $\rho_i \subset \rho_j$ implies $V(\rho_i) \leq V(\rho_j)$. The monetary function maps ρ_i, of which the value space is discrete and heterogeneous, into V, of which the value space is continuous. Because the privacy set of one user is always changing and the privacy sets of different users at some time are different with each other, this continuous and homogeneous mapping is significant.

For U, the basic incentive of trading privacy is to gain some utility from other aspects; we denote "the other aspect" as the "target goal", and the corresponding gain as the "target gain" T. A large problem is that the consumption of privacy is not necessarily correlated with the gain of the target goal; it is possible that a change in the privacy domain has no influence on the domain of the taget goal. To circumvent this problem, we introduce a intermediate domain, with quantification measurement Δ, to "connect" the privacy domain and the target domain. The intermediate domain is similar to the currency in economics. With the help of it, the consumption and gain can be transformed into:

$$V = \mu(\Delta), T = \nu(\Delta) \tag{2.1}$$

For simplicity, we assume each unit of Δ has a linear relationship with the target gain and the privacy consumption:

$$V = p\Delta, T = t\Delta \tag{2.2}$$

where p is the subjective privacy price of U and t is the marginal target gain.

We assume that U can freely consume any level of its privacy, by associating various data set with various privacy policies. The incentive problem of U is described as:

$$\arg\max_{\Delta} G(\Delta) = u(-V) + E(u(T)) \tag{2.3}$$

where G is the net gain of the trading, u is the utility function mapping monetary value to satisfaction, and E is the expectation function measuring the uncertain outcome. V and T can be replaced by the form in Equation 2.2.

A rational user will always maximize his net gain of trading, so he will always try to locate the optimal privacy trading portfolio. To solve the optimal solution of Equation 2.3 by calculating the first-order condition, we have:

$$pu'(-V) = E[tu'(T)] \tag{2.4}$$

$$p = E[\frac{u'(T)}{u'(-V)}t] = E[\frac{u'(T)}{u'(-V)}]E(t) + cov(\frac{u'(T)}{u'(-V)}, t) \tag{2.5}$$

The information contained in the Equation 2.4 is important: $E[tu'(T)]$ is the absolute marginal increase of the target utility gain while $pu'(-V)$ is the marginal loss in utility of U; U will always continue to trade privacy for the target goal until the marginal gain equals the marginal loss. Equation 2.5 is even more useful: the second term, $cov(\frac{u'(T)}{u'(-V)}, t)$, is named as *risk adjustment* in economics. If it is zero, the subjective price equals to the one conducted by a rational user without risk concerns; if it is positive, the subjective price will be higher than the risk-neutral result, and if it is negative the subjective price will be lower thus the user might love to take the risk. According to this criterion, we have the following definition:

Definition 2 *(Privacy risk sensitivity) The covariance term in Equation 2.5 quantifies the risk sensitivity regarding to trading privacy. If the covariance is positive, the user is defined as privacy-risk-aversion; if the covariance is negative, the user is defined as privacy-risk-loving; and if the covariance* *is zero, the user is defined as privacy-risk-neutral.*

In cloud computing, the privacy trading mechanism is not only about affording the user's privacy values, but also about compensating his aversion of risks. Even we have introduced a trusted platform of trading, the cloud user may still doubt on the later usage of the secrets. An incentive compatible privacy trading mechanism should take this aspect into consideration.

In real world, the variables related to the target goal, T and t, are always pre-determined by the application scheme. So according to Equation 2.2, Δ can be determined. Suppose U's utility function is $u(x) = ln(x)$, then Equation 2.3 can be transformed to:

$$G(\Delta) = ln(T) - ln(V) = ln(\frac{t}{p}) = ln(\frac{t}{p^\star + \delta}) \tag{2.6}$$

where $p^\star$ is the marginal loss of privacy calculated by a rational privacy-risk-neutral cloud user, and δ is the risk adjustment quantified by the covariance in Equation 2.5.

Theorem 1 *In the cloud platform with transparent and fair trading mechanism of privacy, the cloud users will accept the privacy transaction if and only if the marginal return, t, exceeds the subjective marginal loss, which is the objective marginal loss $p^\star$ plus the subjective risk evaluation δ.*

To make the privacy trading incentive-compatible, or to encourage the risk-aversion user to accept the transactions, the cloud provider should compensate the user. For example, the cloud provider can allocate more resource (either temporal or spatial) to the user, valued as ξ. If the cloud provider by chance decide to compensate in the domain satisfying the user's target goal, Equation 2.6 can be further transformed to:

$$G(\Delta) = ln(T) - ln(V) = ln(\frac{t+\xi}{p}) = ln(\frac{t+\xi}{p^\star + \delta}) \tag{2.7}$$

Table 2.1: Specifications of execution time measurement.

Hardware platform	
Processor	AMD Phenom(tm)II X6 1035T Processor 2.60 GHz
Memory Size	8.00 GB
Operating System	Windows 7 64-bit
Software specification	
BIOS	SeaBIOS 1.7.0
Simulator	QEMU 1.0.1
Cloud user workload	
Operation	N tasks
Task	100 circles
Interval to call NeuCloud	η tasks
Percentage of tasks to in NeuCloud	ζ

Theorem 2 *In order to make the privacy trading incentive-compatible, the cloud provider should compensate the user with the value* $\xi = p^{\star} + \delta - t$.

The specific estimation of $p^{\star}$, δ and t is an empirical study and we will leave it to the future work.

2.5 Implementation and Evaluation

The hardware and software specifications of our implementation are listed in Table 2.1. We build a BIOS with modified NeuCloud SMM handlers based on the open-source SeaBIOS project, and use the QEMU as the simulator. We set the user space workload as an operation with N tasks. Each task consists of 100 circles of execution (in our experiment, for simplicity, each execution is an empty loop written in assembly language). NeuCloud (SMM mode) is called every η tasks in the normal system mode, and the total percentage of tasks executed in NeuCloud is ζ.

Figure 2.10 shows the relationship between the execution time and the interval to call NeuCloud, when $N = 100000$ and $\zeta = 50\%$. In the figure, $\eta = 5$ means every 5 tasks executed in normal system mode, NeuCloud is called and there are 5 tasks to be executed in NeuCloud (since $\zeta = 50\%$, the numbers

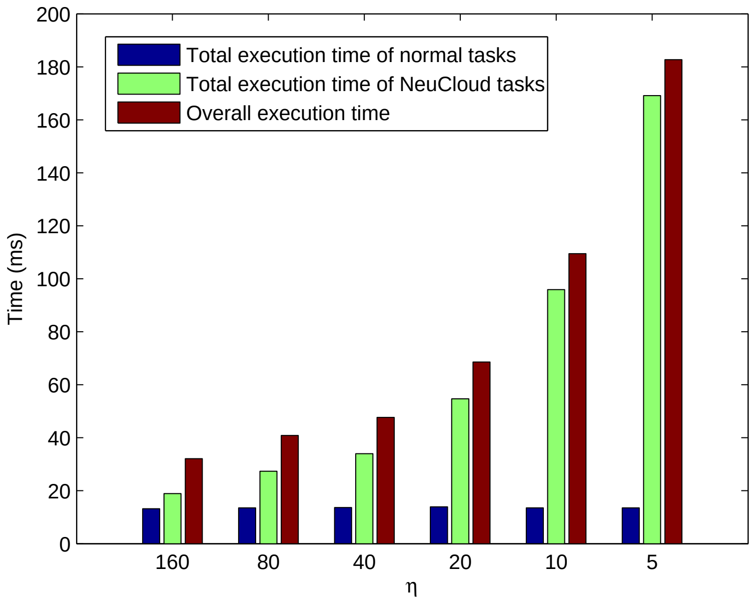


Figure 2.10: The relationship between the execution time and the interval to call NeuCloud ($N = 10000, \zeta = 50\%$).

of tasks outside and inside NeuCloud are the same). From the results we can learn that frequently calling NeuCloud would lead to a significant increase of overhead and the frequency has an exponential relationship with the overhead. That indicates that we should minimize the frequency to call NeuCloud by accumulating tasks and executing them all at once in NeuCloud.

Figure 2.11 shows the relationship between the execution time and the percentage of tasks executed in NeuCloud, from which we can learn that 1) execution in SMM is way faster than in normal system mode; 2) the more tasks executed in NeuCloud, the faster the operation will be finished.

2.6 A Variation of NeuCloud

The NeuCloud architecture set up a concrete base for trusted and transparent privacy trading. However, it does have several drawbacks:

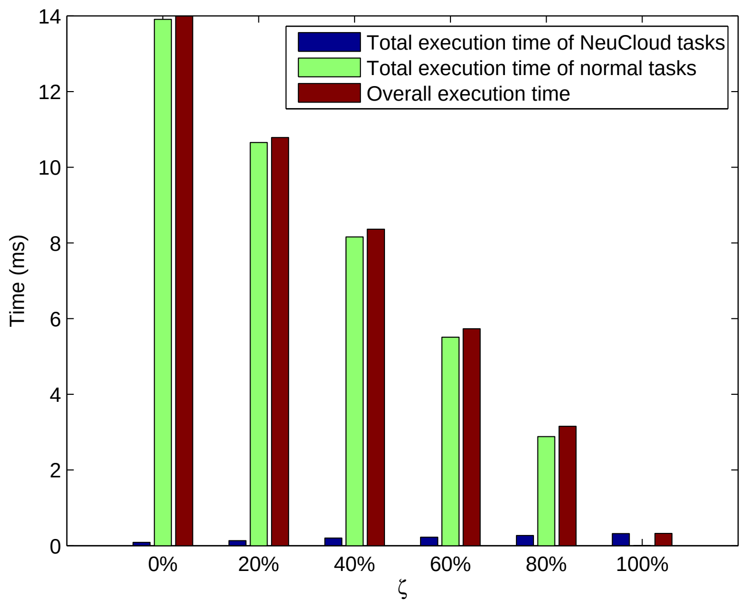


Figure 2.11: The relationship between the execution time and the percentage of tasks executed in NeuCloud ($N = 10000$).

1. **Scheduling complexity**: when the processor jumps into SMM, all the current tasks in the normal system mode will be interrupted and paused, until the processor's returning back from SMM. So the scheduling design should be pretty tricky: 1) we can't trap into SMM as soon as we meet a NeuCloud call, because according to the results in Figure 2.10, the high frequency will kill the system performance; 2) we may accumulate a bunch of NeuCloud call and execute them at one time, and the results shown in Figure 2.11 have proved that it won't take a long time to execute in SMM. But the threshold of accumulated task number is not easy to set, since the waiting time in the queue is also performance killer.

2. **Storage protection**: SMM disables all the interruptions, which is both good and bad. The good side is that no one can interrupt the execution in the protected domain. But the bad side is that NeuCloud cannot control any hardware devices with the help of interruption, for example, hard drive read/write. Thus so far NeuCloud only provides memory protection, without storage protection.

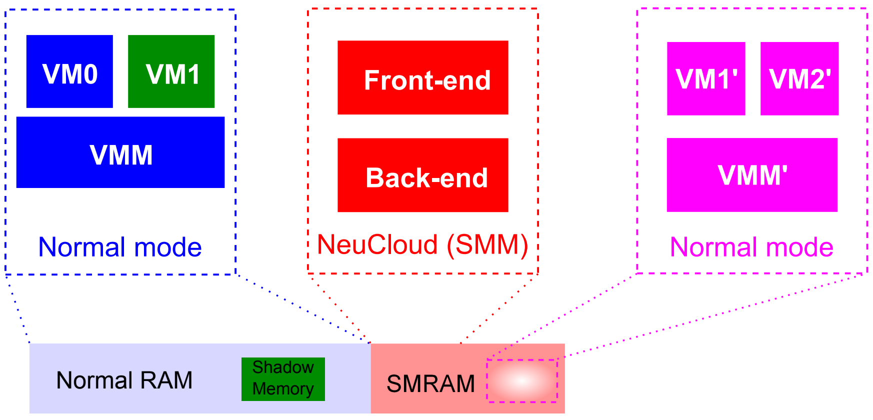


Figure 2.12: A variation of NeuCloud: the architecture.

Figure 2.12 illustrates the improved version of NeuCloud. When NeuCloud is called, it no longer directly executes any tasks. Instead, it shrinks the SMRAM region and “exposes” a VMM in normal system mode to run the tasks. After tasks are finished, the new VMM turns back control to NeuCloud and then NeuCloud returns to the legacy VMM. The new VMM being summoned by NeuCloud is also pre-programmed into the BIOS along with the other NeuCloud code. Because the summoned VMM is executed in normal system mode, the above drawbacks can be well solved. Figure 2.13 illustrates the implementation of this improved variation of NeuCloud.

1. After CRTM, BIOS loads NeuCloud code into SMRAM, and set the SMRAM range (up to 4GB).

2. BIOS loads the MBR of the NeuCloud VMM to the hard drive. If the boot-loader exceeds 512 Byte, the NeuCloud VMM MBR should boot a stage-1 block outside MBR to perform the future loading work.

3. The NeuCloud VMM MBR first of all shrink back the SMRAM, then boot the NeuCloud VMM. We intentionally locate the VMM memory space inside the SMRAM. After booting up, the NeuCloud VMM returns to the bootloader, which reset the SMRAM to “cover” the NeuCloud VMM into protection.

4. Then the NeuCloud VMM MBR loads the legacy VMM MBR from the disk.

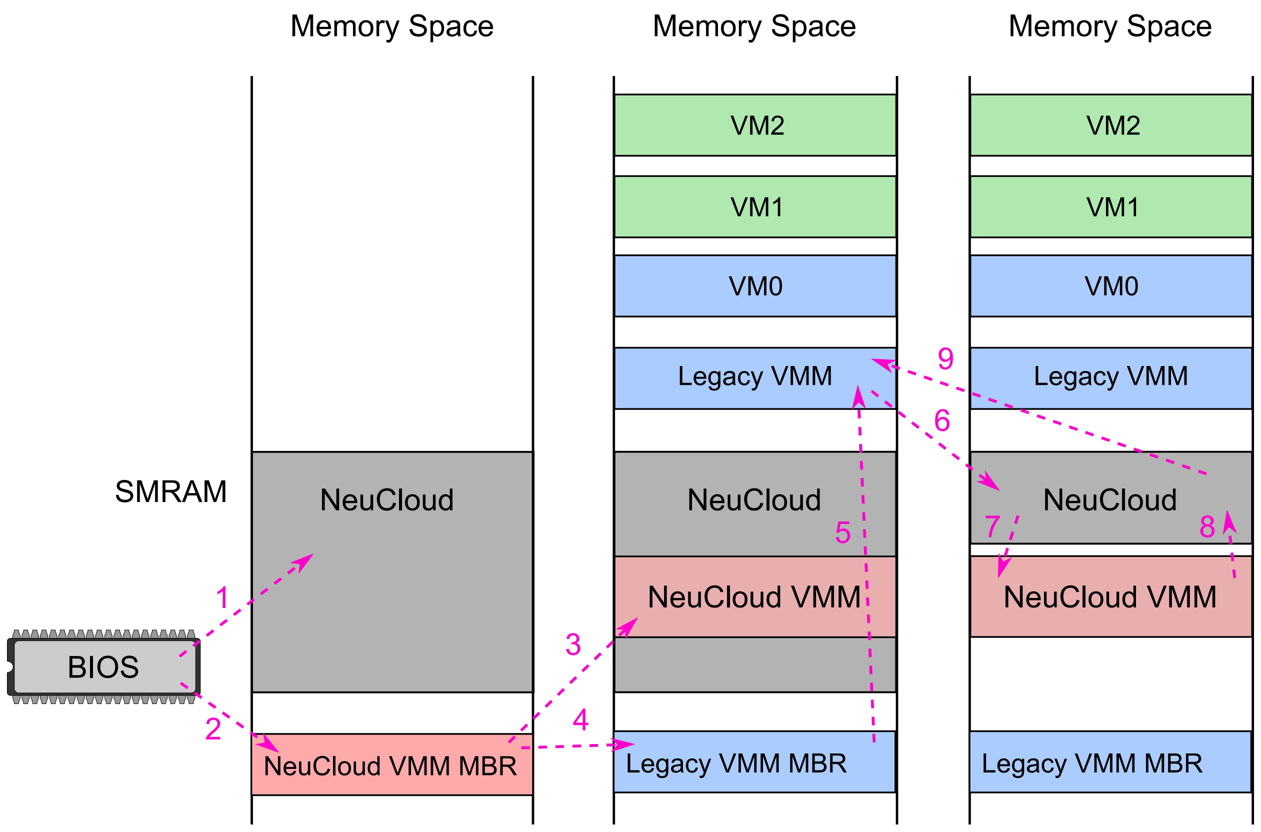


Figure 2.13: A variation of NeuCloud: the implementation

5. From now on, everything is the same with the legacy platform. The legacy VMM is loaded and then the VMs.

6. When NeuCloud is needed, the legacy VMM requests a SMI.

7. Upon entering into SMM, NeuCloud front-end works as we described before.

8. Then NeuCloud shrinks back the SMRAM and releases the NeuCloud VMM to serve the workload.

9. Once finishing the tasks, NeuCloud turns control back to NeuCloud by calling SMI.

10. NeuCloud reset the SMRAM to cover the NeuCloud VMM, and returns to the legacy VMM.

Due to time and space limitations, we will evaluate the variation of NeuCloud in our future work.

2.7 Summary

In this chapter, we proposes a novel architecture, NeuCloud, to enable partially, trusted, transparently, accountably privacy manipulation and revelation. With the help of this architecture, the privacy-sensitive users can be more confident to move to public clouds. A trusted computing base is not enough, in order to stimulate incentive-compatible privacy trading, we present a theoretical framework and provide the guidelines for cloud provider to compensate the cloud user's privacy-risk-aversion. We implement the NeuCloud and evaluate it. Moreover, a improved model of NeuCloud is discussed.

Chapter 3

No free-riders: Incentive Compatible Moving Target Defense Against VM-Colocation Attacks in Clouds

Although IaaS offers cost-efficiency and ease-of-use to cloud users, there are significant security concerns that need to be addressed when considering moving critical applications and sensitive data to the clouds. Recent work [41] reveals the problem of side-channel based attacks through virtual machine colocation, showing that the adversaries can map the internal VM-placement of the cloud and mount cross-VM side-channel attacks by placing malicious VMs on the victim's physical server.

Many software level approaches [37] and hardware modification methods [47, 26] have been developed against the side-channel attacks. Unfortunately, the software approach cannot perfectly cover the new advances in attack models, and the hardware approach, modifying the cloud physical platform, is far from practical for commercial clouds. Using commercial off-the-shelf (COTS) components, the *Shamir's secret sharing* approach [43] can make the attack much harder. We can split our secret *D*

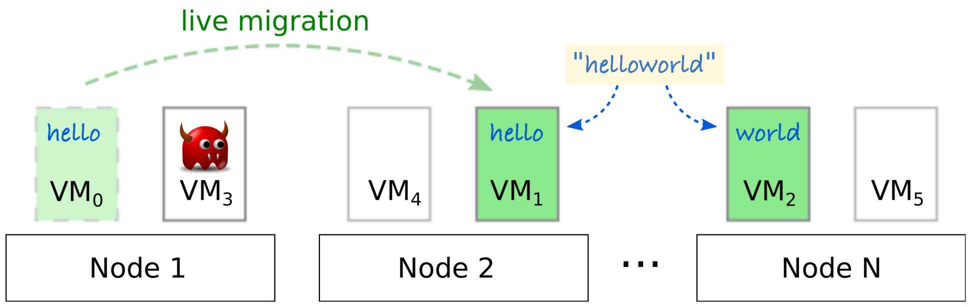


Figure 3.1: Shamir's secret sharing in cloud environment and the moving target defense through VM-migration.

into k pieces and store them into k VMs [1]. Using Shamir's secret sharing, D can be easily constructed from all k pieces but knowledge less than k pieces reveals no information about D. Note that the secret sharing might be controlled by either a single party or multiple parties [6]. Thus, the attacker will be forced to use *brute-force-attacks* to achieve colocation with multiple target VMs, according to [41]. Now, the defense problems becomes how to protect the k pieces from being all captured by the attacker.

The methods to securely live-migrate VMs [14, 30] can make it much harder for adversaries to locate the target VMs [24]. As shown in Figure 3.1, the secret, "helloworld", is divided into two pieces and held by two VMs, initially as VM_0 and VM_2. An attacker can launch VMs colocated with the benign VMs with some probabilities. For example, if VM_3 is controlled by an attacker, the attacker will be able to extract partial secret, "hello", out of VM_0. Although the splitting of "helloworld" into "hello" and "world" can prevent the attack from reconstructing the secret from VM_3, we should at the same time guarantee that VM_5 can never be taken over by the same attacker. The fact that the attack would simultaneously attempt to launch lots of VMs to enumerate the colocations with both VM_0 and VM_2 motivates the migration of the VMs, e.g. moving VM_0 to VM_1. Unfortunately, there is currently no formal model to guide the dynamic migration strategy for clouds. Similarly, quantifying the benefits of dynamism still remains an open problem.

Intuitively, the cloud provider can offload the choice to cloud users, letting them migrating at free will.

[1]It is not necessary to have k identical VMs. We can have one service VM and $k-1$ secret holding VMs that can be very small and cost little.

Or the cloud provider can migrate VMs according to the security demand - those VMs with higher security demand of dynamism have higher chances to be migrated. At first glance, this is reasonable. However, as long as some of the VMs sharing the secret are migrated, other VMs can take the free ride of the achieved dynamism. Since any rational cloud user will try to maximize their benefit[2] , he/she could report a lower preference of dynamism than the actual one, pining its hope on other VMs honest movements to reach the security goal. So in this case, the game equilibrium is that no one would truthfully report the security valuation and migrate.

Our goal and contributions: To solve the aforementioned problems, the goals of this work are (1) to model the migration benefit and cost, (2) to provide instructional criteria for the migration strategy of the cloud provider, and (3) to motivate the cloud users to migrate in a incentive-compatible way. In order to address these issues, we develop a migration strategy based on the Vickrey-Clarke-Groves(VCG) mechanism[36] in game theories, which can maximize the social welfare given the individuals are all "selfish".

The contributions of this work are as follows. To the best of our knowledge, (1) this is the first work to address VM-colocation based attacks in clouds using moving target defense; (2) our work is the first effort to apply VCG game and realize incentive compatible migration in cloud; (3) we offer two criteria about how to make the strategy acceptable by rational cloud users, and how to determine the optimal time interval of migrations, (4) our analytical evaluation shows that our defense approach is practical for commercial clouds.

This chapter is organized as follows. In Section 3.1, we review some of the fundamental concepts of game theory, especially the VCG mechanism. In Section 3.2, we present our proposed method. In Section 3.3, we evaluate the security and practicability of our scheme. We conclude the work in Section 3.4.

[2]Note that even if all the VMs sharing the secret are controlled by one user, because different VMs may have different purposes, each VM should be treated as an independent party with individual interest.

3.1 Preliminaries

3.1.1 Assumptions and Notations

We have the following assumptions: (1) the cloud controller and the migration process are all secure; (2) for simplicity, each migration of a VM will result in a constant cost in terms of service interruption and consume the same amount of resources; (3) the cloud provider has enough CPU, network bandwidth, and other resources to perform arbitrary migration; (4) the cloud provider has sufficient resources as the reward, e.g., extra memory or CPUs, to motivate the players to migrate, which will be further discussed later; and (5) a node's destination of migration is randomly chosen, as long as the cloud has free space. In reality, cloud provider would take performance and efficacy into consideration during the migration. For example, VMs with frequent connections should be placed close to each other. In this work, for simplicity, we only measure security in the goal function, but a commercial cloud can freely modify the goal function as needed, where our game model remains the same.

The incentive problem of cloud migration can be modelled as a game. The specific game discussed here is a finite player, single round, simultaneous action, incomplete information game, with payoffs depending on the final result of players' actions. In the cloud migration problem, the game players, $P_1, P_2, ..., P_n$, are n VMs sharing a secret. The cloud provider (game mediator) is denoted as P_t, who doesn't participate in the game but operates the game. Players have actions which they can perform at designated times in the game, and as a result they receive payoffs. The actions of the players are denoted as $X = (x_1, x_2, ..., x_n)$, and specially $X_{-i} = (x_1, x_2, ..., x_{i-1}, x_{i+1}, ..., x_n)$. The players have different pieces of information, on which the payoffs may depend; each player has his/her individual strategy to maximize his or her payoff. Each player $P_i \in \{P_1, P_2, ..., P_n\}$ can decide whether to accept the migration request from P_t or not, so $x_i = \{accept, refuse\}$. For those who agree to migrate, the direct payment is $\bar{p}_i$, which measures the cost of downtime due to live migration. Here we assume the cost of downtime is a constant value for each P_i. As a result of migration, all the players get benefited.

We measure the benefit as $v_i(X)$, where the valuation function v_i quantifies P_i's real security demand of the overall dynamism. However, under the assumption of incomplete information, P_i can decide to report a fake valuation function v_i' at will. Finally, the utility function of each player is $u_i = v_i - p_i$, where p_i is the total payment.

3.1.2 Incentive Compatible Game

While our ultimate goal is to design a migration strategy to fulfil some security requirements, the problem is that in real-world cloud environment most of the cloud users are not willing to migrate their VMs unless it is necessary, because the migration will result in service interruption. So the main purpose of our work is to design an incentive compatible mechanism to mitigate this problem. Following Nisan's work [36], the terms "mechanism" and "incentive compatible" are defined as :

Definition 3 *(Mechanism) [36] Given a set of n players, and a set of outcomes, A, let V_i be the set of possible valuation functions of the form $v_i(a)$ which player i could have for an outcome $a \in A$. A mechanism is a function $f : V_1 \times V_2 \times ... \times V_n \to A$. Given the valuations claimed by the players, f selects an outcome, and n payment functions, $p_1, p_2, ..., p_n$, where $p_i : V_1 \times V_2 \times ... \times V_n \to R$.*

Definition 4 *(Incentive compatible) [36] If, for every player i, every $v_1 \in V_1, v_2 \in V_2, ..., v_n \in V_n$, and every $v_i' \in V_i$, where $a = f(v_i, v_{-i})$ and $a' = f(v_i', v_{-i})$, then $v_i(a) - p_i(v_i, v_{-i}) \geq v_i(a') - p_i(v_i', v_{-i})$, then the mechanism is incentive compatible.*

Specially, among those incentive compatible mechanisms, the Vickrey-Clarke-Groves (VCG) mechanism is the mostly used one. The VCG mechanism generally seeks to maximize the social welfare of all players in one game, where the social welfare is calculated as $\sum_{i=1}^{n} v_i$. So the goal function of VCG is $argmax_{a \in A} \sum_{i=1}^{n} v_i$. The VCG mechanism and the rule to design VCG mechanisms [36] are defined as:

Definition 5 *(Vickrey-Clarke-Groves mechanism) [36] A mechanism, consisting of payment functions $p_1, p_2, ..., p_n$ and a function f, for a game with outcome set A, is a Vickrey-Clarke-Groves mechanism if $f(v_1, v_2, ..., v_n) = argmax_{a \in A} \sum v_i(a)$ (f maximizes the social welfare) and for some functions $h_1, h_2, ..., h_n$, where $h_i : V_{-i} \rightarrow R$ (h_i does not depend on v_i), $\forall v_i \in V, p_i(v_i) = h(v_{-i}) - \sum_{j \neq i} v_j$.*

Definition 6 *(Clarke pivot rule) [36] The choice $h_i(v_{-i}) = max_{b \in A} \sum_{j \neq i} v_i(b)$ is called the Clarke pivot payment. Under this rule the payment of player i is $p_i(v_1, v_2, ..., v_n) = max_b \sum_{j \neq i} v_i(b) - \sum_{j \neq i} v_i(a)$, where $a = f(v_1, v_2, ..., v_n)$.*

3.2 VM-migration Based Moving Target Defence

3.2.1 The Optimal Number of Moving VMs

At first glance, it appears that maximum dynamism can be achieved if all VMs migrate, but in this section we will disprove it and find the optimal number of moving VMs. As denoted in Section 3.1, each player has the unique valuation function $v_i(X)$, where $X = (x_1, x_2, ..., x_n)$ is the vector of actions of all players. Intuitively, v_i has a positive correlation with the randomness or diversity of the VM placement. Suppose there are m physical nodes as the available candidates, and γ of them are randomly chosen as the migration destination, of which the capacities (the maximum number of extra VMs one node can host using its current free space) are $C = c_1, c_2, ..., c_\gamma$. If k out of n VMs are randomly selected to be migrated ($k < \sum_{i=1}^{\gamma} c_i$), the number of possible placements, $N(m, \gamma, n, k, C)$ can be derived using the *Balls In Bins* analysis.

For selecting γ ones out of the m nodes, and selecting k ones out of the n VMs, the numbers of possible schemes are given respectively:

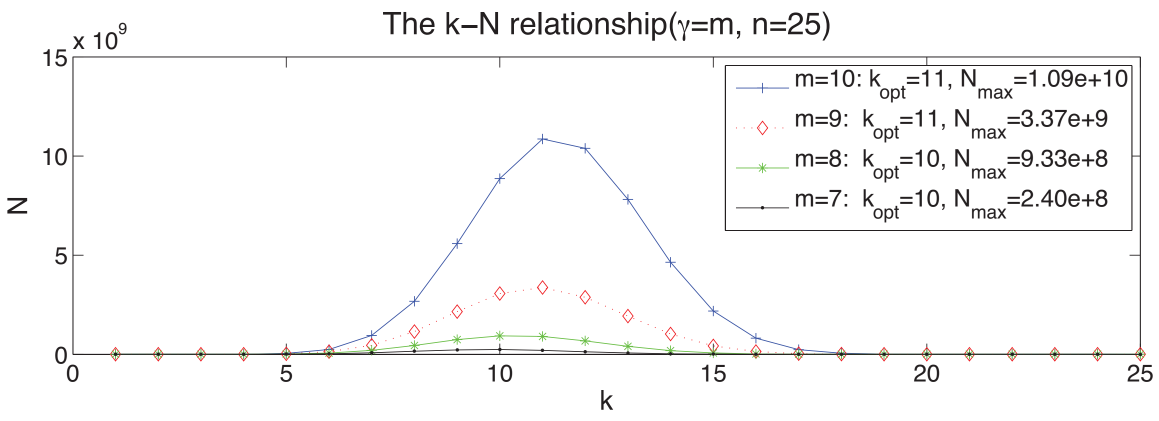


Figure 3.2: The k-N relationship when n=25, $\bar{c} = 4$, and m=γ varies from 10 to 7.

$$S(m,\gamma) = \binom{m}{\gamma}, T(n,k) = \binom{n}{k} \tag{3.1}$$

The generating function for placing k distinguishable VMs (balls) in γ distinguishable nodes (bins) is

$$GF = \prod_{i=1}^{\gamma} \sum_{j=0}^{c_i} \frac{x^j}{j!} \tag{3.2}$$

where the coefficient of $\frac{x^k}{k!}$, denoted as $\theta(\gamma, k, C)$, is the number of the total number of possible placements.

Specially, if $c_1 = c_2 = ... = c_\gamma = \bar{c}$, Equation 3.2 reduces to be

$$GF = (1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + ... + \frac{x^{\bar{c}}}{\bar{c}!})^\gamma \tag{3.3}$$

Finally, the total number of possible VM-placements is

$$N(m,\gamma,n,k,C) = S(m,\gamma)T(n,k)\theta(\gamma,k,C) \tag{3.4}$$

Given the m, γ, n and C, we can use Equation 3.4 to find k_{opt}, which is the optimal number of moving VMs corresponding to the largest N. To make it tangible, an example curve illustrating the $k - N$ relationship is shown in Figure 3.2. Since $S(m, \gamma)$ does not contribute too much in the magnitude of N, we simply set $\gamma = m$ (so $S(m, \gamma) = 1$); for demonstration, we set the total VMs as $n = 25$ and the capacity of each destination node as $\bar{c} = 4$. From the curve we can learn that:

1. It is not true that a larger k is better, which means that the intuitive strategy of migrating VMs as many as possible is incorrect.

2. A larger γ, namely more destination nodes, significantly increases the number of possible placement schemes.

3. Compared to γ, N is even more sensitive to k. N can reach the order of 10^9 when k is still in the order of 10^1, which implies the huge potential of VM migration as a moving target defense strategy.

On the one hand, the optimal k is given by

$$k_{opt} = argmax_k N(m, \gamma, n, k, C) \tag{3.5}$$

and in reality, only cloud provider controls the complete information of m, γ, n, k, C, so it is the cloud provider's responsibility to calculate the k_{opt}.

On the other hand, if we assign 1 to *accept* and 0 to *refuse*, we have the actual number of moving VMs, k_{real}, as:

$$k_{real} = \sum_{i=1}^{n} x_i \tag{3.6}$$

In economics, the valuation function should reflect the security preference of the cloud users. Since this is not the main focus of this work, we simply model the preference as a linear function of the system diversity:

$$v_i(X)|_{m,\gamma,n,C} = \lambda_i N(m, \gamma, n, \sum_{i=1}^{n} x_i, C) + \eta_i \tag{3.7}$$

where λ_i and η_i are determined by the ith cloud user's preference.

Once the cloud provider decides the security level, and thus the value of k, the rest is to motivate the randomly chosen k VMs, denoted as Ω, to get migrated. To satisfy the Clarke pivot rule defined in Definition 6, we design the payment function for each $P_i \in \Omega$ as:

$$p_i(X) = \sum_{j \neq i} v_j(X_{-i}) - \sum_{j \neq i} v_j(X) + \bar{p}_i \tag{3.8}$$

where $\bar{p}_i$ is the constant cost due to service interruption as assumed. We will further introduce the determination of $\bar{p}_i$ in Section 3.2.2.

The second part of Equation 3.8, $-\sum_{j \neq i} v_j(X)$, is actually a positive reward from cloud provider to the ith VM. It can be any form with monetary value equalling to it, like extra CPU scheduling credit, bonus VM life, extra network bandwidth and so on, depending on the need of P_i. The first part, $\sum_{j \neq i} v_j(X_{-i})$, is the monetary charge of P_i. If P_i chooses *accept*, $\bar{p}_i$ is the actual migration cost; if P_i chooses *refuse*, $\bar{p}_i$ is in the form of monetary charge. Given that everyone else accepts migration, if P_i decides to migrate, we have $k_{real} = k_{opt}$; if P_i refuses to do so, $k_{real} = k_{opt} - 1$.

Theorem 3 *(The criterion for the migration decision) The mechanism with the valuation function shown in Equation 3.7 and the payment function shown in Equation 3.8 is individually rational if and only if*

$$\sum_{i=1}^{n} \lambda_i (N(m, \gamma, n, k_{opt}, C) - N(m, \gamma, n, k_{opt} - 1, C)) > \bar{p}_i$$

Proof: To prove that the mechanism is individually rational, we should show that the mechanism has a utility of at least zero for each of the players. For those selected to be migrated:

$$\begin{aligned} u_i(X) = v_i(X) - p_i(X) &= \sum_{i=1}^{n} v_i(X) - \sum_{j\neq i}^{n} v_j(X_{-i}) - \bar{p}_i \\ &= \sum_{i=1}^{n} \lambda_i (N(m,\gamma,n,k_{opt},C) - N(m,\gamma,n,k_{opt}-1,C)) - \bar{p}_i \end{aligned} \tag{3.9}$$

Only when the value of the above equation is larger than 0, P_i can achieve individual rationality. As a matter of fact, Theorem 3 provides a criterion for cloud provider to decide whether a migration is cost-efficient.

Theorem 4 *(Incentive Compatibility) The mechanism with the above valuation and payment functions is incentive compatible and thus motivates the players to truthfully reveal their security preference.*

Proof: To get to the proof of incentive compatibility, we must show that for any given i, X_{-i} and the P_i's choice $x_i = accept$ or $x_i = refuse$:

$$u_i(X = accept \cup X_{-i}) \geq u_i(X' = refuse \cup X_{-i}) \tag{3.10}$$

The utility of P_i for X is given by

$$u_i(X) = v_i(X) - p_i(X) = \sum_{i=1}^{n} v_i(X) - \sum_{j\neq i}^{n} v_j(X_{-i}) - \bar{p}_i \tag{3.11}$$

Likewise,

$$u_i(X') = \sum_{i=1}^{n} v_i(X') - \sum_{j\neq i}^{n} v_j(X_{-i}) - \bar{p}_i \tag{3.12}$$

So we have

Table 3.1: Platform specifications of migration downtime experiment.

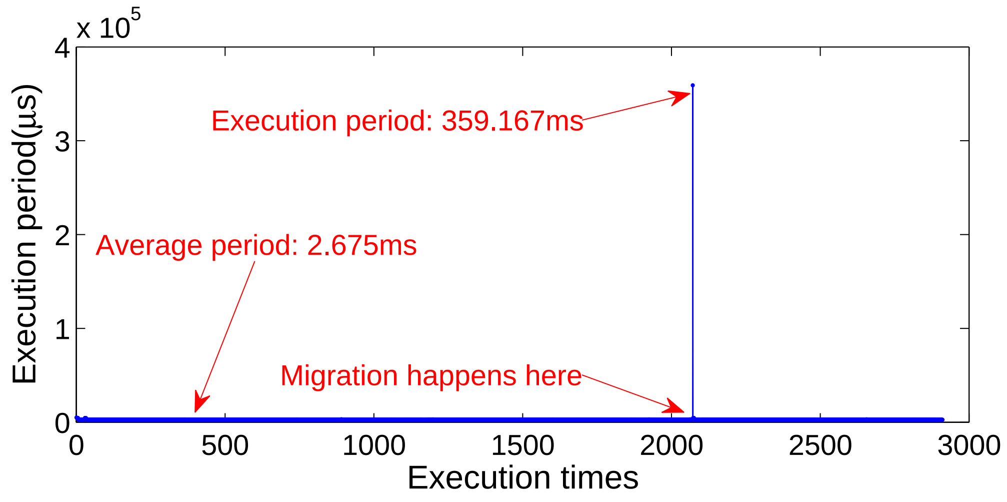
Hardware (source and destination nodes):	**VMM:**
CPU: Intel Xeon x5650 2.66GHz × 2	Xen version: 4.0.1-21326-02-0.5
RAM: 8GB DDR3 1333MHz × 3	Dom0: openSUSE 11.3 x86_64
Ethernet: Broadcom NetXtreme II BCM5709	Dom0 kernel: 2.6.34.7-0.7-xen

$$\begin{aligned} u_i(X) - u_i(X') &= \sum_{i=1}^{n} v_i(X) - \sum_{i=1}^{n} v_i(X') \\ &= \sum_{i=1}^{n} \lambda_i(N(m,\gamma,n,k_{opt},C) - N(m,\gamma,n,k_{opt}-1,C)) \end{aligned} \tag{3.13}$$

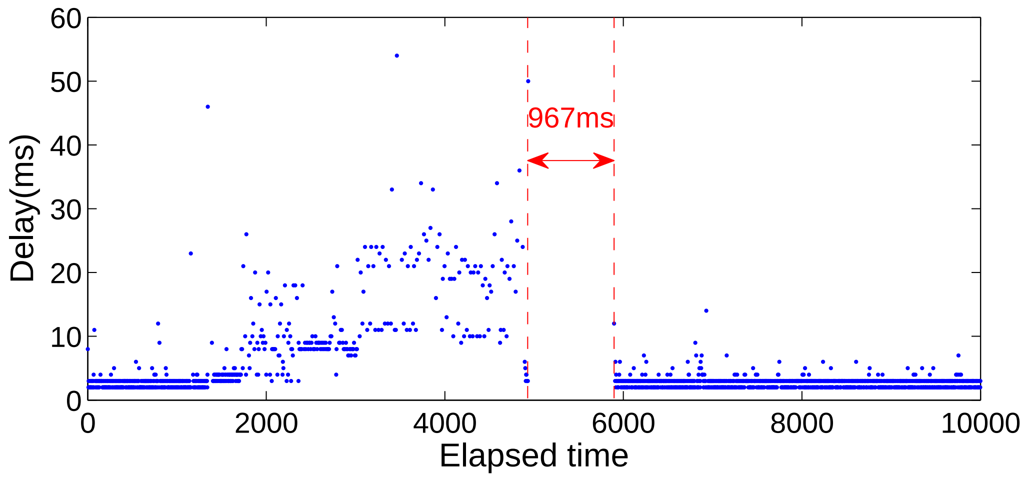
According to Equation 3.5, the value of Equation 3.13 is always non-negative. Therefore, the mechanism is incentive compatible.

3.2.2 The Constant Cost of Migration

In Equation 3.8 we introduced the $\bar{p}_i$, which is the constant cost of service interruption each time P_i is migrated. According to [30], $\bar{p}_i$ is determined by the initial memory size of a VM and the applications' memory access pattern. For demonstration, we illustrate the downtime due to migration in Figure 3.3. The platform specifications are listed in Table 3.1, and the VM that we migrate is of 1 vcpu, 1 GB memory and 4 GB disk. To test the influence to the cpu execution, we keep the VM executing "`for(i=0;i<999999;i++); gettimeofday(&time,NULL);`" and recording the execution period. As shown in 3.3a, the migration will bring in an execution delay of around 350ms. To test the influence to the network service, we keep measuring the VM's response delay of `GET` requests. As shown in 3.3b, there will be a 967ms downtime to the web server. P_t and each P_i can estimate the $\bar{p}_i$ respectively, according to the service downtime.



(a) Impact on CPU execution time of a certain task due to migration. As illustrated in the graph, the execution downtime is around 360ms.



(b) Migration impact on response delay of web server. As illustrated in the graph, the web service downtime due to migration is 967ms.

Figure 3.3: The influence of migration to VM computation and web service.

3.2.3 Defense Timeline

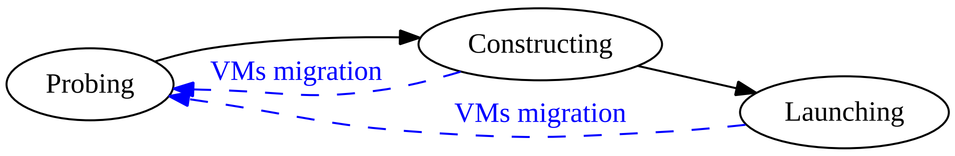
Although the migration can confuse the attackers and force them to restart the attack procedures, with time going by the attackers can still figure out the new VM placement as well. To solve the problem, we have to keep migrating the VMs, and how to determine the optimal time interval becomes a new problem. Following the model in [16], the State Transition Diagram (STG) of attacks is shown in Figure 3.4a. More specifically, the attackers have to firstly probe the placement of the VMs, for example, by continually creating and stopping their probing VMs and testing the colocation with target VMs; then, it takes some time to construct the attacks; and there is still a period before the attackers successfully gathering all the information. Any distortion of the VM placement, for example, by migrating any of the VMs to other places, will reset the attack to the initial state.

As shown in Figure 3.4b, suppose the total time needed for a successful attack is t_1, the time interval between any two migrations t_2 should satisfy $t_2 < t_1$, which offers the cloud provider the criterion to determine migration intervals. If $t_2 \geq t_1$, there is a highly chance for the attackers to fully extract the secret. In reality the constructing time and the launching time of the attacks could be treated as constants, while the probing time has a linear relationship with the N, which explodes with the increasing of γ as shown in Figure 3.5d.

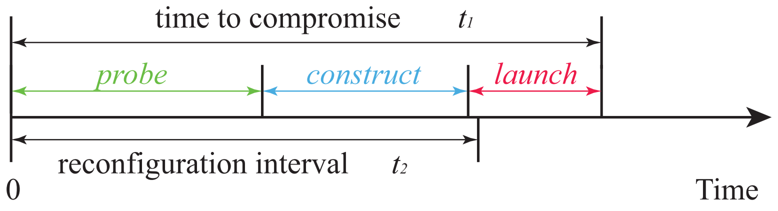
3.3 Evaluation

3.3.1 Security

We can never guarantee that our defense strategy or algorithm is agnostic to the adversaries. So when we design a moving target defense scheme, we must ensure that the adversaries are impossible or difficult to recover our actual internal infrastructures even if the strategy is not secret. In the mechanism



(a) The State Transition Diagram (STG) of attacks.



(b) Attack and defense lifetime.

Figure 3.4: The timeline of attacks and moving target defense (VM migration).

designed here, we assume that the forms of the valuation function and the deliberate payment function are known to the public. However, in reality the individual security preference is kept as a privacy of each cloud user, thus the adversaries have no way to get λ_i and η_i. Even if some of these parameter-pairs are exposed, as long as the adversaries cannot obtain the information of all the VMs in the cloud, our strategy based on the overall social welfare is always secure. Furthermore, during the generation of the migration scheme, γ are randomly chosen and the capacity $C = \{c_1, c_2, ..., c_n\}$ is a secret of cloud provider only, thus according to Equation 3.5 the final migration scheme is only known to cloud provider.

Another indicator of one moving target defense scheme's capability is the number of the total target space. As demonstrated later in Figure 3.5d, N can increase rapidly with a small increase of γ. With the exploding number of possible VM placements, the adversaries are extremely difficult to enumerate all the possibilities.

We note that there might be information leakage during VM migration. But since the design of migration methods is not within the scope of this work, we prospect more secured VM migration mechanisms.

Table 3.2: Amazon EC2 Linux/UNIX instances pricing and specifications (US East, up to January 2012) [2] .

Instance types	Price	Memory	Compute Units	Storage	API name
Small	\$0.085 per hour	1.7 GB	1	160 GB	m1.small
Large	\$0.34 per hour	7.5 GB	4	850 GB	m1.large

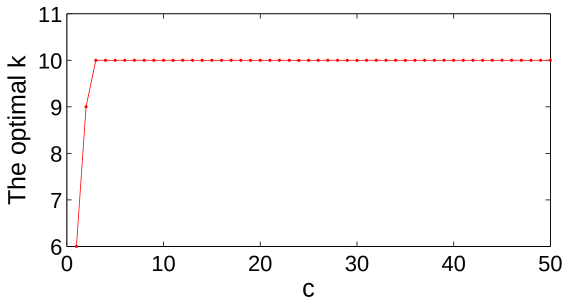
3.3.2 Practicability

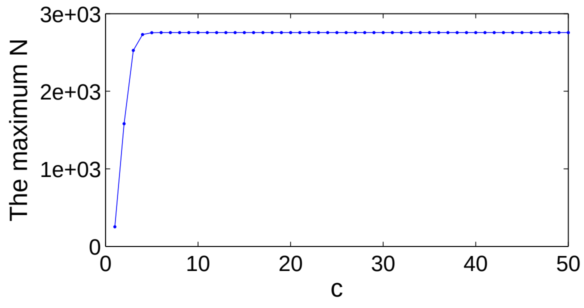
As shown in Figure 3.5, compared with $\bar{c}$, γ contributes more on the value of both k_{opt} and N_{max}. Empirically, $k_{opt} \approx \gamma$; theoretically, for any $c_1, c_2, x \in \mathbb{N}$ *and* $c_1 < c_2$ we have $\frac{x^{c_1}}{c_1!} > \frac{x^{c_2}}{c_2!}$ and $lim_{c\to\infty}\frac{x^c}{c!} = 0$, so according to Equation 3.3, γ has a significantly larger influence on the value of k_{opt}. Consequently, for the real-world cloud providers, it is pretty easy to determine the optimal strategy by assigning $k_{opt} = \gamma$ without complex algorithms, which maks this approach scalable.

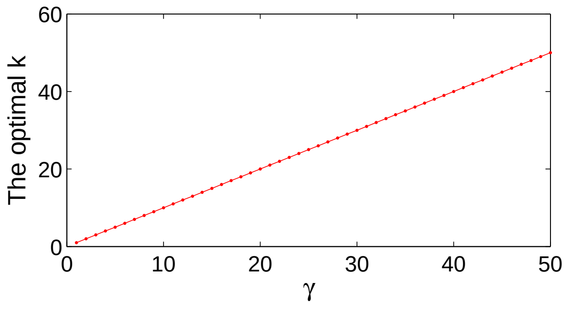
Another potential concern is that whether the cloud users are willing to split a secret into multiple VMs, which is the precondition of our VM migration based defense. As shown in Table 3.2, the small instance has a price of $1/4$ of the price of the large one, and offers $1/4$ of the compute units, $1/4.4$ of the memory and $1/5.3$ of the storage. Therefore splitting a secret from one large VM into four samll VMs won't largely increase the cost. Compare with the risk of being attacked through covert channels, the cloud users will be willing to endure the cost due to transferring computation from few large VMs to numerous small VMs.

3.4 Summary

To summarize, we proposed an incentive compatible moving target defense of cloud VM-colocation attacks, based on the VCG mechanism. When the migration is acceptable by rational users and how to determine the time interval are discussed. Our analysis shows that this defense strategy is practical to be applied to commercial clouds.


(a) The $\bar{c} - k_{opt}$ relationship ($m = \gamma = 10$).


(b) The $\bar{c} - N_{max}$ relationship ($m = \gamma = 10$).


(c) The $\gamma - k_{opt}$ relationship ($\bar{c} = 4$).

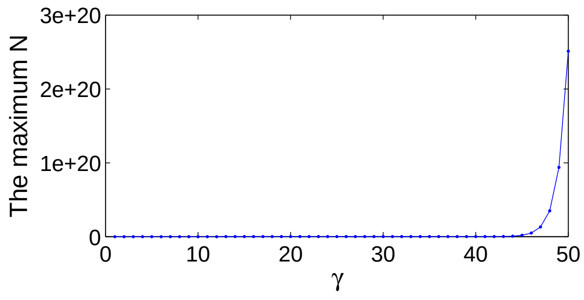

(d) The $\gamma - N_{max}$ relationship ($\bar{c} = 4$).

Figure 3.5: The influence to k_{opt} and corresponding N_{max} by adjusting $\bar{c}$ or γ in Equation 3.5.

Chapter 4

Conclusion and Future Works

Cloud computing has proved itself as one of the greatest inventions in human history, but the current design brings in some troublesome problems, like the balance of cooperation benefits and privacy concerns between the cloud provider and the cloud users, and the balance of cooperation benefits and free-rider concerns between different cloud users. This thesis strives to address theses two incentive problems. To solve the service-privacy conflict, a novel cloud computing architecture is proposed enabling trusted and transparent privacy exchange. With the trusted cloud provider as the arbiter, the free-rider problem also gets settled based on a game theory model.

The solution to the first problem can be further improved. As described in Section 2.6, we can ameliorate the drawbacks of NeuCloud, such as scheduling complicacy and device management restrictions, by inserting a new VMM context into the architecture. Moreover, although NeuCloud sheds a light on trusted and transparent privacy trading, some crucial factors in the privacy trading theories need future empirical study and investigation.

The second part of work can also be extended in different avenues. First, only security goal is considered here for simplicity, but the valuation and payment functions can be easily adjusted to include the performance considerations. Second, so far we have only considered the non-cooperative game. Actually, there might be some connections between VMs on cloud, forming a cooperative game which is a competition between coalitions of players rather than between individual players. In the future, we will develop the mechanism in regard to the coordination game. Furthermore, in the next step, we will extend the mechanism to suite into the asymmetric information game, because in reality some VMs are offering services to other VMs (thus knowing some statistics of the latter), and some VMs may even be controlled by attackers with some knowledge of other VMs.

As a whole, the main objective of this study is to provide an incentive-compatible to eliminate the cloud user's privacy or cooperative concerns. We hope that the proposed methodology can benefit those commercial clouds some day. The theoretical part of this work can be extended to other fields where incentive concerns about privacy, trust and cooperation exist.

Bibliography

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

[2] Amazon web services. http://aws.amazon.com

[3] Eucalyptus cloud. http://www.eucalyptus.com

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

[5] Opennebula. http://opennebula.org

[6] Workshop on Secret Sharing and Cloud Computing. MEXT, Institute of Mathematics for Industry, Kyushu University, Japan (2011)

[7] Armbrust, M., Fox, A., Griffith, R., Joseph, A.D., Katz, R.H., Konwinski, A., Lee, G., Patterson, D.A., Rabkin, A., Stoica, I., Zaharia, M.: Above the clouds: A berkeley view of cloud computing. Tech. Rep. UCB/EECS-2009-28, EECS Department, University of California, Berkeley (Feb 2009), http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.html

[8] Azab, A.M., Ning, P., Wang, Z., Jiang, X., Zhang, X., Skalsky, N.C.: Hypersentry: enabling stealthy in-context measurement of hypervisor integrity. In: Proceedings of the 17th ACM con-

ference on Computer and communications security. pp. 38–49. CCS '10, ACM, New York, NY, USA (2010), http://doi.acm.org/10.1145/1866307.1866313

[9] Azab, A.M., Ning, P., Zhang, X.: Sice: a hardware-level strongly isolated computing environment for x86 multi-core platforms. In: Proceedings of the 18th ACM conference on Computer and communications security. pp. 375–388. CCS '11, ACM, New York, NY, USA (2011), http://doi.acm.org/10.1145/2046707.2046752

[10] Barham, P., Dragovic, B., Fraser, K., Hand, S., Harris, T., Ho, A., Neugebauer, R., Pratt, I., Warfield, A.: Xen and the art of virtualization. SIGOPS Oper. Syst. Rev. 37, 164–177 (October 2003), http://doi.acm.org/10.1145/1165389.945462

[11] Berger, S., Cáceres, R., Goldman, K.A., Perez, R., Sailer, R., Doorn, L.: vtpm: Virtualizing the trusted platform module. In: In USENIX Security. pp. 305–320 (2006)

[12] Cerbelaud, D., Garg, S., Huylebroeck, J.: Opening the clouds: qualitative overview of the state-of-the-art open source vm-based cloud management platforms. In: Proceedings of the 10th ACM/IFIP/USENIX International Conference on Middleware. pp. 22:1–22:8. Middleware '09, Springer-Verlag New York, Inc., New York, NY, USA (2009), http://portal.acm.org/citation.cfm?id=1656980.1657011

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

[14] Danev, B., Masti, R.J., Karame, G.O., Capkun, S.: Enabling secure vm-vtpm migration in private clouds. In: Proceedings of the 27th Annual Computer Security Applications Conference. pp. 187–196. ACSAC '11, ACM, New York, NY, USA (2011)

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

[16] Evans, D., Nguyen-Tuong, A., Knight, J.: Effectiveness of moving target defenses. In: Jajodia, S., Ghosh, A.K., Swarup, V., Wang, C., Wang, X.S. (eds.) Moving Target Defense, Advances in Information Security, vol. 54, pp. 29–48. Springer New York (2011)

[17] Garfinkel, T., Pfaff, B., Chow, J., Rosenblum, M., Boneh, D.: Terra: a virtual machine-based platform for trusted computing. SIGOPS Oper. Syst. Rev. 37, 193–206 (October 2003), `http://doi.acm.org/10.1145/1165389.945464`

[18] Gellman, R.: Privacy in the clouds: Risks to privacy and confidentiality from cloud computing. Tech. rep., World Privacy Forum (2009)

[19] Gentry, C.: Computing arbitrary functions of encrypted data. Commun. ACM 53, 97–105 (March 2010), `http://doi.acm.org/10.1145/1666420.1666444`

[20] Habib, I.: Virtualization with kvm. Linux J. 2008 (February 2008), `http://portal.acm.org/citation.cfm?id=1344209.1344217`

[21] Intel Corporation: Intel® 64 and IA-32 Architectures Software Developer's Manual. No. 253669-033US (December 2009)

[22] Jiang Wang, A.S., Ghosh, A.: Hypercheck: A hardware-assisted integrity monitor. Tech. Rep. GMU-CS-TR-2010-5, Department of Computer Science, George Mason University, 4400 University Drive MSN 4A5, Fairfax, VA 22030-4444 USA (2010)

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

[24] Kant, K.: Configuration management security in data center environments. In: Jajodia, S., Ghosh, A.K., Swarup, V., Wang, C., Wang, X.S. (eds.) Moving Target Defense, Advances in Information Security, vol. 54, pp. 161–181. Springer New York (2011)

[25] Keller, E., Szefer, J., Rexford, J., Lee, R.B.: Nohype: virtualized cloud infrastructure without the virtualization. SIGARCH Comput. Archit. News 38, 350–361 (June 2010), `http://doi.acm.org/10.1145/1816038.1816010`

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

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

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

[29] Li, J., Wang, Q., Wang, C., Cao, N., Ren, K., Lou, W.: Fuzzy keyword search over encrypted data in cloud computing. In: INFOCOM, 2010 Proceedings IEEE. pp. 1 –5 (march 2010)

[30] Liu, H., Xu, C.Z., Jin, H., Gong, J., Liao, X.: Performance and energy modeling for live migration of virtual machines. In: Proceedings of the 20th international symposium on High performance distributed computing. pp. 171–182. HPDC '11, ACM, New York, NY, USA (2011)

[31] Liu, Q., Wang, G., Wu, J.: An efficient privacy preserving keyword search scheme in cloud computing. In: Computational Science and Engineering, 2009. CSE '09. International Conference on. vol. 2, pp. 715 –720 (aug 2009)

[32] Mayes, K.E., Markantonakis, K., Tomlinson, A.: Introduction to the tpm. In: Smart Cards, Tokens, Security and Applications, pp. 155–172. Springer US (2008)

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

[34] Murray, D.G., Milos, G., Hand, S.: Improving xen security through disaggregation. In: Proceedings of the fourth ACM SIGPLAN/SIGOPS international conference on Virtual execution environments. pp. 151–160. VEE '08, ACM, New York, NY, USA (2008), http://doi.acm.org/10.1145/1346256.1346278

[35] Nance, K., Bishop, M., Hay, B.: Virtual machine introspection: Observation or interference? IEEE Security and Privacy 6, 32–37 (September 2008), http://portal.acm.org/citation.cfm?id=1477059.1477286

[36] Nisan, N., Roughgarden, T., Tardos, E., Vazirani, V.: Introduction to Mechanism Design (for Computer Scientists). Cambridge University Press (2007)

[37] Page, D.: Defending against cache-based side-channel attacks. Information Security Technical Report 8(1), 30 – 44 (2003)

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

[39] Pearson, S., Shen, Y., Mowbray, M.: A privacy manager for cloud computing. In: Proceedings of the 1st International Conference on Cloud Computing. pp. 90–106. CloudCom '09, Springer-Verlag, Berlin, Heidelberg (2009), http://dx.doi.org/10.1007/978-3-642-10665-1_9

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

[41] Ristenpart, T., Tromer, E., Shacham, H., Savage, S.: 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. pp. 199–212. CCS '09, ACM, New York, NY, USA (2009)

[42] Ristenpart, T., Tromer, E., Shacham, H., Savage, S.: 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. pp. 199–212. CCS '09, ACM, New York, NY, USA (2009), `http://doi.acm.org/10.1145/1653662.1653687`

[43] Shamir, A.: How to share a secret. Commun. ACM 22, 612–613 (November 1979)

[44] Smith, S.W., Safford, D.: Practical server privacy with secure coprocessors. IBM Systems Journal 40(3), 683 –695 (2001)

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

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

[47] Wang, Z., Lee, R.B.: New cache designs for thwarting software cache-based side channel attacks. SIGARCH Comput. Archit. News 35, 494–505 (Jun 2007)

Vita

Yulong Zhang was born on January 10th, 1989, in Changsha, China and is a Chinese Citizen. In 2006, he received his Bachelor of Science in Microelectronics and double major in Economics from Peking University, Beijing, China. His research interests include privacy protection in cloud computing, hardware and hypervisor architecture design for virtualization, and applications of game and microeconomics theories in cyber-security.

Experience:

Research Assistant: Virginia Commonwealth University, Richmond, VA (2011–Present)

Research Assistant: Peking University, Beijing, China (2008–2010)

Intern Software Engineer: Chinese Software Development Lab of IBM, Beijing, China (2010)

Publications:

1. Y.L. Zhang, M. Li, K. Bai, M. Yu, and W.Y. Zang. Incentive Compatible Moving Target Defense against VM-Colocation Attacks in Clouds. In IFIP International Information Security and Privacy Conference 2012, Heraklion, Crete, Greece, 4-6 June 2012.

2. W.Q. Pan, Y.L. Zhang, M. Yu and J.J. Wu. Improving Virtualization Security by Splitting Hypervisor into Smaller Components. Submitted to 26th Annual WG 11.3 Conference on Data and Applications Security and Privacy, Paris, France. July, 2012.

3. Y.L. Zhang, G.C. Sun, W.G. Wu. Diode Characteristic of Electrolyte-oxide-semiconductor Structure for Potential Chemical and Biological Applications. In Solid-State Sensors, Actuators and Microsystems Conference, Beijing, China, June 2011.

4. Y.L. Zhang, H.Y. Mao, W.G. Wu. Fabrication of Orderly and Uniform Nanotip Arrays Based on Oxygen Plasma Etching of Photoresist. In 11th Annual Conference of the China Society of Micro-Nano Technology, Harbin, China, Oct. 2009.

5. H.Y. Mao, W.G. Wu, Y.L. Zhang, G. Zhai and J. Xu, "Fabrication of high-compact nanowires using alternating photoresist ashing and spacer technology", Journal of Micromechanics and Microengineering, Vol 20, 8, 2010.

6. H.Y. Mao, Y.L. Zhang, W.G. Wu. Fabrication of nanofiber-based SERS-active substrates by oxygen plasma removal of SU-8 photoresist. In 5th IEEE International Conference of Nano/Micro Engineered and Molecular Systems, Xiamen, China, Jan. 2010.

7. C. Qian, C. Ni, Y.L. Zhang, Y. Zhou, W.G. Wu, J. Xu, And Y.L. Hao. A Highly-Ordered Three-Dimensional Petal-Like Arrayed Structure for Highly Surface-Enhanced Raman Scattering. In 23rd IEEE International Conference on Micro Electro Mechanical Systems, Hong Kong, China, Jan. 2010.

8. H.Y. Mao, W.G. Wu, Q.H. Liu, Y.L. Zhang and Y. Li. Nanofiber-based Surface Microfluidic Structures for Cell and Nanoparticle Patterning. In 14th International Conference on Miniaturized Systems for Chemistry and Life Sciences, Groningen, Netherlands, Oct. 2010.

9. M. Li, Y.L. Zhang, K. Bai, M. Yu, W.Y. Zang and X.B. He. Improving Cloud Survivability through Dependency based Virtual Machine Placement. Submitted to International Conference on Security and Cryptography 2012. (Submitted)