RepCloud: Achieving Fine-grained Cloud TCB Attestation with Reputation Systems

Anbang Ruan
Department of Computer Science
University of Oxford
anbang.ruan@cs.ox.ac.uk

Andrew Martin
Department of Computer Science
University of Oxford
andrew.martin@cs.ox.ac.uk

ABSTRACT

Security concerns for emerging cloud computing models have become the focus of much research, but little of this targets the underlying infrastructure. Trusted Cloud proposals generally assert that the Trusted Computing Base (TCB) of the cloud should be clearly defined and attested to. However, specific characteristics of trust in the cloud make such solutions difficult to implement in an effective and practical way. We present RepCloud, a reputation system for managing decentralized attestation metrics in the cloud. We observe that as being deterministic and tamper-proof, trust evidence generated by the TCG framework can be efficiently transmitted within the cloud. In a web of nodes with high connectivity and mutual-attestation frequency, corrupted nodes can be identified effectively. By modelling this web with RepCloud, we achieved a fine-grained cloud TCB attestation scheme with high confidence for trust. Cloud users can determine the security properties of the exact nodes that may affect the genuine functionalities of their applications, without obtaining much internal information of the cloud. Experiments showed that besides achieved fine-grained attestation RepCloud still incurred lower trust management overhead than existing trusted cloud proposals.

Categories and Subject Descriptors

D.4.6 [**Operating Systems**]: Security and Protection

General Terms

Security

Keywords

Remote Attestation, Trusted Cloud, Reputation System

1. INTRODUCTION

Substantial security threats to the rapidly-growing cloud computing service model are attracting a great deal of attention. The spectre of massive data loss has drawn cloud users to pay serious attention to the confidentiality and integrity of their data and computation when these are outsourced to a cloud provider. Various cloud security mechanisms have been proposed to protect users' benefits with different perspectives. Authorization and authentication mechanisms protect users' assets in the cloud from unauthorized accesses. Auditing modules record actions performed in the cloud for evaluation at a future point in time. IDS and firewall protect internal cloud resource from attacks, and VMI (Virtual Machine Introspection) interfaces [10] enable fine-grained inner-VM (Virtual Machine) protections from security services provided by the cloud, e.g. antivirus and rootkit detector. A full line of work for secure data outsourcing has also been proposed. Furthermore, Cloud Security Alliance [1] defines a comprehensive cloud security reference model, with twelve security domains covering different level of security mechanisms in the cloud.

However, most of these cloud security mechanisms target threats against the higher layer components in the cloud model, while their functionalities still rely on the integrity of lower supporting layers, i.e. their Trusted Computing Base (TCB) should still be clearly identified and attested to. For example, in IaaS (Infrastructure as a Service) cloud systems, security modules, such as audit or access control, can be easily circumvented when their underlying virtualization layer has been tampered with [10]. VMI can also be abused to leak users' privacy. Moreover, as in the cloud computing paradigm, the infrastructure no longer belongs to the end users. It is hence necessary for the cloud providers to generate evidence to their users for verifying that the services and security mechanisms they advertised are actually being enforced, and the first step among which is to prove the trustworthiness of the cloud TCB. As a party cannot simply claim its own trustworthiness [17], a Trusted Third Party (TTP) should be introduced for generating this evidence.

The TCG (Trusted Computing Group) [4] proposes to attach a platform with a TPM (Trusted Platform Module), which can genuinely record and then report the platform's configuration. The tamper-proof nature of the TPM and its implanted cryptographic protocols empower it to act as a TTP dedicated to this platform. The trust evidence it generates can hence be used by users to perform *remote attestations* [4] to the platform to determine its genuine configuration. Accordingly, several trusted IaaS cloud systems have been proposed to integrate critical hardware resources (which we refer to as *nodes*) with TPMs and provided services for the users to remotely attest to either the entire cloud (which we refer to as *combined attestation*) [22, 15] or the specific nodes hosting their VMs (the *separated attestation*) [23]. However, we argue that effective and practical trust management for users' *cloud applications* (i.e. set of VMs implementing users' tasks) cannot be achieved without clearly defining their TCBs, which requires the cloud's specific characteristics to be considered:

Complexity. Typical IaaS cloud systems may usually have a huge amount of nodes, but only a small part of them will take part in the life cycle of all the VMs of a user's cloud application. As we will discuss in Section 3, this set of concerning nodes (i.e. the *dependency* of the application) usually contains not only the nodes hosting the VMs, but also various internal facilities, e.g. the management nodes and the security service modules introduced above. It is hence difficult for general users to identify all of them for the separated attestation. Attack surface may also be widened when the internal infrastructure of the cloud is exposed. On the other hand, for the combined attestation, evidence for the trustworthiness of the entire cloud is usually too coarse-grained for users to be convinced, e.g. a single certificate declaring that the entire Amazon Web Service (AWS) [6] is trustworthy. Moreover, when the states of all nodes in the cloud are considered altogether, fault in any may result in the entire cloud to appear as compromised.

Dynamics. TCG trust evidence (or *attestation tickets*) can only indicate the trustworthiness of a platform up to the time when it is generated. As the state of a node changes from time to time, attestations should be performed repeatedly. Moreover, as VMs can be migrated among different nodes, and these migrations are ought to be transparent to users, difficulties are intensified for separated attestation scheme to accurately determine the trust states of the exact dependency for a user's cloud application. On the other hand, as appropriate repeatedly attestation timing can only be decided according to the communication patterns of particular VMs, it is hence difficult for the combined attestation scheme to determine the strategies to satisfy attestation needs of every application in the cloud.

Heterogeneity. To satisfy different security requirements, various security modules are implemented inside different parts of a cloud. A same module can also expose varying properties in the view of different applications when considering its policy configurations. These modules can hence result in diversities in the properties of the cloud TCB regarding different applications. For example, the traffic among the VMs of an application are monitored by an internal firewall, while another application uses VMI-based anti-virus module provided by the cloud [14]. However, it is difficult for users to determine the properties of the dependency of their applications by merely getting the knowledge of the configurations of the entire cloud. It is also hard for the separated scheme to identify these properties especially when considering the complexity and dynamics of the cloud.

We observe that, though a cloud is usually organized in a centralized way, the communication patterns inside are actually distributed. Trust states regarding a cloud application can hence be managed in a decentralized way to achieve fine-grained and dynamic cloud TCB identification and attestation. In this paper, we propose RepCloud, a reputation system for managing *TCG trust* (i.e. trust relationship maintained by the TCG infrastructure) in an IaaS cloud. Reputation systems [11] have been proposed for determining and disseminating trust in distributed applications with highly interconnected nodes, e.g. P2P systems. Trust towards a node is evaluated separately by all the nodes it has interacted with, and is aggregated to form a global representation. In a web of nodes that has a high level of connectivity and interaction frequency, corrupted nodes can be efficiently detected. As nodes in a cloud share a similar interaction pattern with these systems, reputation systems can be introduced for aggregating and disseminating the TCG trust. With the *deterministic* and *tamper-proof* nature of the TCG trust evidence, trust in the cloud can be managed in an effective and practical way.

In RepCloud, *decentralized local attestations* are performed among nodes inside the cloud based on their interactions. Trust evidence gathered by nodes is disseminated among corresponding nodes to reduce redundant attestations. It is also aggregated to form a global *web-of-trust*, with which a higher level of trust semantics can be deduced. We made the following contributions:

1) ***Fine-grained cloud TCB definition.*** We identified the TCB of an application in an IaaS cloud, with the scale of the concerning nodes minimized. We observed that by monitoring the direct communication among nodes and performing decentralized attestation, the TCB can be dynamically determined and with its trust state continuously updated. Fine-grained attestations to cloud applications can hence be achieved, with which users can determine the security properties of the exact nodes that may affect the genuine functionalities of their VMs.

2) ***Aggregation and dissemination of TCG trust.*** We approached the first step towards integrating the Trusted Computing framework with reputations systems. We managed trust in the cloud in a decentralized and autonomous way, and proposed to aggregate and disseminate the TCG trust evidence. Our experiments show that, with reputation systems, TCG trust is effectively and efficiently transmitted among interconnecting nodes. Besides, redundant attestations are dramatically reduced, while still maintaining a low level of state-change-detection delay.

3) ***Confidence of trust and cloud TCB attestation.*** We proposed the concept of *Confidence of Trust (COT)*, as a representation of the reputation of a node in the cloud, for modeling the attestation relationship deduced from the global web-of-trust. COT value for a node represents its trustworthiness and indicates a consistent trend in its attestations. Hence, continuous trustworthiness can be determined from the discrete trust evidence generated by the TCG framework. We presented a new paradigm for implementing the cloud TCB attestation takes into consideration the COT and the TCG trust evidence.

Section 2 introduces the background and our motivations. Section 3 presents our cloud model and analyses the adversary behaviors. The definitions for the cloud TCB are then proposed. In Section 4, we illustrate the RepCloud framework for gathering and disseminating TCG trust. We propose the calculations for the COT and preliminary cloud TCB attestation procedures in Section 5. Our experiments and implementation are presented in Section 6 and Section 7 respectively. Section 8 presents the discussion to our work, and Section 9 summarizes related work. Finally, we conclude our paper and discuss future work in Section 10.

2. BACKGROUND AND MOTIVATION

Trusted Computing. Trusted Computing Group (TCG)

[4] specifies a hardware module, the Trusted Platform Module (TPM), to serve as a root of trust for a platform. The attestation architecture of TCG aims at *measuring* all loaded executables by hashing each piece of software into the TPM before loading it. The platform is bootstrapped by the CRTM (the Core Root of Trust for Measurement), which is trusted by default and will initiate measurements to the BIOS and the boot loader to construct a *chain of trust*. This chain is then extended through every operating system components up to the applications and their configuration files. Once the executables are measured into the TPM, the TPM can reliably attest to the hash values by signing them with a TPM protected key, i.e. the *AIK* (Attestation Identity Key). The signed values (i.e. the *attestation tickets*) are sent to the challenger (or verifier), who can then decide whether to trust the target platform. The tickets generated by the TCG framework are ***deterministic***, i.e. they can reveal the genuine state of the target platform, as they are based on the tamper-proof registers protected by the TPM. Meanwhile, the tickets are themselves ***tamper-proof***, as they are signed by the TPM protected key. However, these tickets are ***transient*** as they can only reflect the genuine state of the platform up to the time when they are generated. Hence, attestations should be performed regularly for updating the trust states.

As with the limited computation capability of a TPM chip, a complete attestation procedure between two nodes may require several seconds, the latency of which is unacceptable when attestation is performed regularly. Stumpf et al. [24] proposed the Timestamped Hashchain-based Attestation to compensate this deficiency. To avoid performing expensive TPM operations for every attestation request, the server (the node to be attested to) uses a nonce issued by a Trusted Third Party (TTP) binding to the timestamp of the current time to perform the TPM_Quote instruction regularly and generate an attestation ticket each time. As long as the time have been synchronized, the client can determine the trustworthiness of the server from the ticket and deduce the freshness of the attestation from the nonce. As the nonce is generated from a global time instead of randomly chosen by the client, attestation tickets generated by this scheme are hence ***disseminable*** and can be reused by a third party, e.g. different clients.

Reputation System. In many collaborative applications such as P2P systems, trust only represents a "personal" view of a peer, and reputation systems [11, 13] are used for a peer to infer trust towards a stranger by consulting other peers it trusts or by aggregating all the others' views, in assumption that most peers will "tell the truth". A typical reputation system usually defines following components [11]: 1) *formulation,* the mathematical presentation of the reputation metric, together with the sources of input to that formulation; 2) *calculation,* the algorithm to calculate the formulation for a given set of constraints; 3) *dissemination,* the mechanism for storing and disseminating the reputation value. As this "personal" view is very easy to forge, various attacks to the reputation systems exist, while several countermeasures have also been proposed [11].

Motivation. As a single attestation ticket can undeniably reveal the genuine trust state of a node, TCG trust can be effectively and efficiently disseminated in the cloud with reputation systems. Hence, by performing attestations among nodes inside the cloud based on their interactions and disseminating the attestation tickets, redundant attestation efforts can be saved. With adaptive specification of re-attest timing, a low average for repeatedly attestations interval can still be achieved. Moreover, from the mutual-attestation relationship among nodes, a global *web-of-trust* can be constructed. We observe that in the web with nodes regularly attesting to each other and exchanging newly generated tickets, corrupted nodes can be quickly identified. With the communication semantics is maintained by the web, the dependency of an application can be easily identified. With additional attestations from a third party to this web, malicious collaborators [11] can also be discovered. Hence, by constructing and modelling this web, we can achieve fine-grained cloud TCB attestation with high confidence for trust. In Section 4, we will present the RepCloud framework for constructing this web-of-trust, and in Section 5, we define the notion of confidence of trust to model this web and present a preliminary attestation procedure.

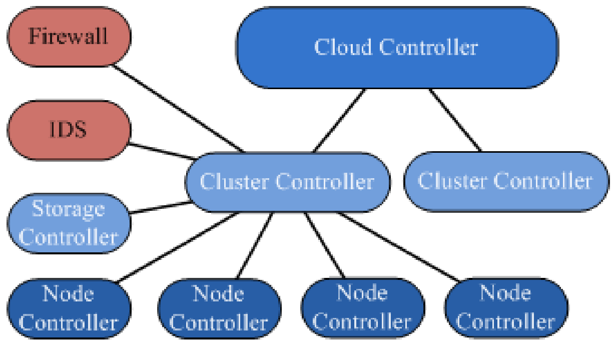


Figure 1: Cloud model

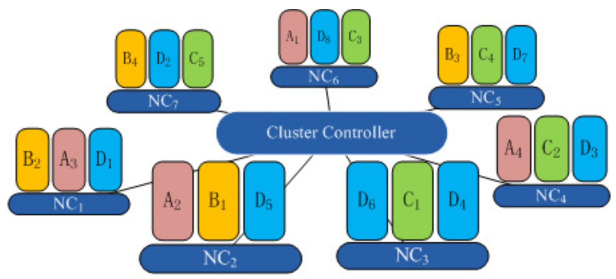
3. CLOUD TCB DEFINITION

In the this section, we will first introduce our cloud model and, correspondingly, the adversary model. The definitions for cloud TCB are then presented.

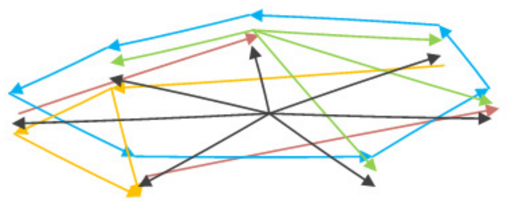
3.1 Cloud Model

Figure 1 depicts the typical IaaS cloud architecture [6], which is used by Amazon, one of the most prevalent IaaS cloud providers. The Node Controllers (NCs) are the basic computing units in the cloud. They host and execute the Virtual Machines (VMs). A cluster of NCs are managed by a Cluster Controller (CLC). The permanent storage of NCs in a cluster are implemented and managed by a Storage Controller (SC). Additional security components can also be implemented by cloud providers, such as firewall and IDS. Finally, a Cloud Controller (CC) manages all the CLCs and exposes interfaces to end users. Therefore, the genuine behaviors of a cloud application depend on the integrity of the NCs hosting all its VMs, the CLCs managing all these NCs, the corresponding security components, and all other supporting nodes, such as SCs and the CC.

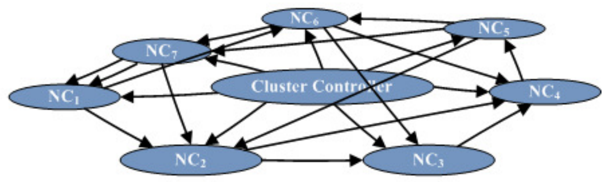
On the other hand, the multi-tenancy nature of an IaaS cloud indicates that VMs belong to different users may share a same node, e.g. the NC. As depicted in Figure 2(a), VMs from different cloud applications scattered among nodes in a cluster. Meanwhile, applications have different internal communication patterns (Fig. 2(b)). For example, in a typical web hosting application, traffic is made frequently from



(a) Cloud applications deployment



(b) Cloud application communication patterns



(c) Aggregated node communication patterns

Figure 2: Cloud communication pattern

the front-end servers to back-end database, while for some MapReduce-based data processing computations, traffic is made from a central master unit to several slave units. The traffic of a single node is hence composed of all the traffic of its VMs, as shown in Figure 2(c).

3.2 Adversary Model

Three types of adversaries are concerned in our cloud model. We differentiate them from the capabilities they can gain in the cloud:

Unprivileged malicious users. Ristenpart et al. [20] presented an attack from unprivileged malicious users towards VMs in the cloud without compromising the hypervisor. Malicious users of this type have the *novel* abilities as the general cloud users. However, new VMs can be instantiated until one is placed co-resident with the target VM. Cross-VM side-channel attacks can then be mounted to extract information from the target VM on the same machine.

Privileged malicious users. Hypervisors are becoming more and more difficult to verify and vulnerable to attacks as their complexity grows [14]. Numerous vulnerabilities have already been shown on the prevalent hypervisors as Xen and VMWare. Attackers need only to gain access to a general VM and run codes that can attack the hypervisor to gain the privileged capabilities. They may then inspect the memory, exposing confidential information or even modify the software running in a VM.

Malicious administrators. Administrators have privileged capabilities in the cloud. They can install or execute malicious programs to perform attacks to users' assets, e.g. to access the memory of a customer's VM. They can also tamper with the management components, including security modules. Policies in the cloud can be modified, secretly violating the SLA (Service Level Agreements).

However, we assume that malicious users and the administrators cannot manipulate the hardware without being identified, i.e. the TPM cannot be tampered with and software components loaded on a TCG-compliant node are genuinely measured. We also do not consider insider physical attacks such as the cold boot attack. As with our cloud model, we assume that direct interactions are needed for the malicious behaviors above to take effect, i.e. attacks can only be launched directly on the target nodes, or on the nodes that have direct interactions with the target nodes.

3.3 TCB Definition

We differentiate TCB definitions regarding three scales of related components:

Core TCB of a node. The functionalities of a VM rely on the underlying virtualization layer, including the hypervisor, the management console and other privileged software modules. Components in this layer also have their dependencies. Hence, as discussed in Section 2, all the components forming the chain-of-trust [4] up to the virtualization layer should be considered. We denote them as the *core TCB of a node*. As some attacks are launched from malicious VMs without changing the core TCB, e.g. side-channel attacks (Sec. 3.2), all the other VMs running on a node should also be considered as in the TCB of the target VM. In this case, we propose to add the countermeasures to the core TCB to diminish these unexpected dependencies among VMs.

Cloud TCB of a node. Nodes in a cloud have direct interactions with others, which we denote as their *neighbours*. VMs on a node may communicate with VMs on the others. The management facilities on a node may also depend on the good behavior of the others, e.g. distributed access control may require modules on different nodes to cooperate and make decisions together. Regarding our adversary model, the *cloud TCB of a node* includes its own core TCB and the core TCBs of all its neighbours. All malicious behaviors to the VMs on a node rely on this scale of TCB.

Cloud TCB of an application. An application in an IaaS cloud contains a set of VMs and possibly several supporting cloud services. Hence, it usually involves interactions among a number of nodes in the cloud. The *cloud TCB of an application* is hence comprised of the cloud TCB of all these nodes. As VMs can be migrated and the location of the supporting services can also be changed accordingly, this TCB dynamically changes during the lifecycle of the application. In this paper, we only concern with the attestation to this TCB at a specific point of time, which has a static set of nodes. By integrating trusted migration protocols [22], attestation to the entire lifecycle can easily be achieved.

4. REPCLOUD FRAMEWORK

In RepCloud, nodes attest to each other based on their interactions. Each node maintains a *Local Trust Vector (LTV)* recording the attestation tickets it gathered by performing attestations to its neighbours. These tickets are then aggregated to construct the global *web-of-trust* to represent the attestation relationship among nodes. This web is maintained by each node in its *Global Trust Metric (GTM)*.

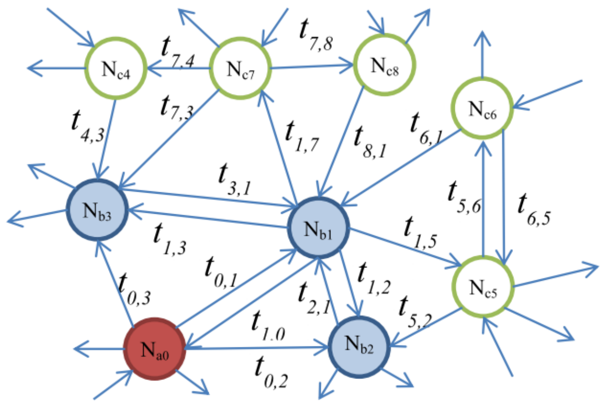


Figure 3: Decentralized local attestation topology.

In this section, we will illustrate RepCloud mechanisms to gather and disseminate the TCG trust. From the web-of-trust, the *attestation history* of each node can be deduced, which records all the attestations performed by other nodes *towards* the node. It provides the source for modelling the confidence of trust in the next section.

4.1 Notations

A node in the cloud is denoted as N_i. The attestation ticket generated by the attestation from N_i towards N_j (performed by N_i) is denoted as:

$$t_{i,j} = (pcr_{i,j}, ts_{i,j})$$

It represents the trust N_i places upon N_j. $pcr_{i,j}$ is the PCR value representing the trust state of N_j, and $ts_{i,j}$ is the timestamp for the time the attestation was performed. The *Local Trust Vector (LTV)* of N_i is hence denoted as:

$$L^i = \vec{t_i}$$

the *Global Trust Metric(GTM)* as:

$$G = [t_{i,j}]$$

We denote the GTM maintained by N_a as G^a. In the following text, we use superscripts only when necessary to denote that the data structure is maintained by the particular node.

As each node only maintains the trust information of its neighbours, only a part of its GTM has valid values. We define $t_{i,j} = (0, 0)$ when no attestation is performed by N_i to N_j. The set Nb^i denotes the neighbours of N_i:

$$Nb^i = \{N_j \mid \forall j(t_{i,j} \neq 0)\}$$

We define ">" for comparing the freshness of the attestation tickets as below. The "max" and "min" are hence self-defined.

$$t_{i,j} > t'_{i,j} \iff ts_{i,j} > ts'_{i,j}$$

Figure 3 illustrates a simplified node attestation topology. Directed edges represent attestations, with their endpoints as the target nodes. Attestation tickets are specified as the weights. For better presentation, nodes attested to by N_{a_0} are referred to as N_{b_i}. They are the neighbours of N_{a_0}, comprising the set Nb^{a_0}. Nodes interacting with N_{b_i} but have no connection to N_{a_0} are depicted as N_{c_j}. With the two-level trust aggregation in RepCloud (Sec. 4.3), all these attestation tickets can be located in the GTM of N_{a_0} (G^{a_0}).

4.2 Local Trust Gathering

A node maintains the trust state of its cloud TCB in its LTV (Local Trust Vector) by performing attestations to its neighbours regularly. The three typical steps of a TCG attestation procedure are considered for gathering local trust:

Measurement. Every node in our cloud model has an embedded TPM, which records the measurement of every component in the trust chain. The entire core TCB is measured. Different runtime attacks exist to inject malicious codes directly into the memory of the node without being measured. However, as countermeasures [18] are being proposed and the TCG attestation framework is also evolving to mitigate these threats, in this paper, we assume that all executables loaded on a node have been genuinely measured, and with the measurement values extended to the TPM.

Attestation. Three types of communication actions with different assumptions on trust are defined in our context: 1) ***Critical communication.*** Security-critical communication actions among nodes require beforehand attestations, because previously generated attestation tickets may be obsolete. These actions can be specified by users, and will trigger local attestation whenever they are encountered. With RepCloud, users can also program to initiate local attestation from a VM to internal nodes in the cloud when necessary. 2) ***General communication.*** For communication actions that do not require immediate attestation, trust evidence can still be gained as proof for service enforcement, e.g. SLA checking or provenance queries [16]. Attestations to a target node in this case can be performed a longer time ago, or by some other nodes that are trusted by the current node. 3) ***Trusted communication.*** Specific communication actions should be performed with pre-assumed trust. Otherwise they may lead to recursive attestations. These actions should be carefully designed and with limited interfaces, e.g. RepCloud protocol communications.

We use the Timestamped Hash-chain attestation protocol [24] in RepCloud to increase the throughput of the attestation operations and generate transmissible attestation tickets. The time of attestation is inferred from the ticket as the nonce is bound to a global time. Every time when an attestation is performed, the PCR value and timestamp will be stored in the LTV of the current node. Every attestation ticket is valid for a specific length of time, after which new attestation should be performed. In RepCloud, as we will discuss in Sec. 6, redundant attestation effort is significantly saved, which least trust management overhead introduced.

Verification. With local attestation, a node is regarded as trustworthy when the measurements of all components in its core TCB are identified in a global expected measurement list. Faulty nodes will be reported. As a node is identified with its AIK (Sec. 7), a new AIK can be issued to the node after it is reset. It is hence regarded as a new node in RepCloud, and its previous distributed maintained attestation information will be discarded automatically when obsolete. All VMs hosted on a faulty node will be halted and the users will be informed. In RepCloud, with attestation to the core TCB, upper layer components can easily be integrated to implement property-based [9, 12] TCB attestation and achieve heterogeneous security property architectures. The core TCB can be extended to contain the software components in the trust chain up to the hypervisor, together with an additional privileged component to introspect the trust states of other privileged components and deduce their properties. These properties can then be disseminated and aggregated with our RepCloud infrastructure. We will propose detailed solutions in our future work.

4.3 Global Trust Aggregation

As a node only concerns the trust state of its neighbours, a local part of the web-of-trust is maintained in its GTM. In this case, a two-level trust aggregation is enough – aggregating the trust that the neighbours' neighbours place towards the neighbours. For example, in Fig. 3, N_{a_0} should obtain all the trust placed upon N_{b_i} by N_{c_j}. In this section, we present the algorithms for constructing the GTM and keeping it from becoming obsolete. In Sec. 6, we will analyse their overhead and discuss methods to minimize it.

4.3.1 *Trust Aggregation*

The GTM of a node is constructed by fetching and merging the LTVs of its neighbours immediately after the attestations to them are performed. The aggregation can be implemented by simply overwriting the corresponding entries in the GTM with those LTVs:

$$G^a_{i,*} = L^i,\ \forall i(N_i \in Nb^a)$$

In this paper, we simplify our cloud model by assuming that nodes tend to interact with their nearby peers, i.e. neighbours of a node also communicate with each other, especially when VMs of a cloud application are more likely to be deployed within a same cluster in the cloud. Hence, the GTM of a node may also contain the trust its neighbours place upon each other. As we can see from Figure 3, by merging the LTV of N_{b_1}, N_{a_0} also gains trust information placed upon its neighbours N_{b_2} ($t_{1,2}$) and N_{b_3} ($t_{1,3}$). When considering scenarios where nodes have little common in communication peers, multi-level of trust aggregation can be used, instead of the two-level in our case. EigenTrust [13] shows that this multi-level aggregation still has a fast convergence. We will extend our model accordingly in our future work.

To obtain more detailed attestation histories, the GTMs of a node's neighbours can be aggregated. To avoid recursively expanding the GTM, a sliced GTM (SGTM), denoted as $SG^{b,a}$, can be returned to N_a to only contain the trust information of all N_a's neighbours maintained by N_b:

$$SG^{b,a} = [(G^b_{i,*})^T]^T \cup [G^b_{*,i}],\ \forall i(N_i \in Nb^a)$$

$SG^{b,a}$ is then extended into G^a. The max operator is used for updating the freshness of the trust information:

$$G^a_{i,j} = \max(G^a_{i,j}, SG^{b,a}_{i,j}),\ \forall b(N_b \in Nb^a)$$

In Figure 3, as N_{a_0} is the neighbours of N_{b_1}, its LTV is maintained in N_{b_1}'s GTM. As N_{b_1} also maintains the LTVs of N_{c_7} and N_{c_5}, $t_{7,3}$ and $t_{5,2}$ are returned in SG^{b_1,a_0}. Moreover, as N_{b_1} is also performing GTM extension, $t_{4,3}$, $t_{8,1}$, and $t_{6,1}$ can also be returned. As the peers are at the same time extending their GTMs, trust is aggregated iteratively.

As defined, $SG^{b,a}$ also contains L^b, when $N_b \in Nb^a$. Hence SGTM can be returned with each attestation instead of LTV. However, in the bootstrapping stage, when a node's GTM is mostly empty, LTV merging can still be used to achieve a faster convergence.

4.3.2 *Trust Dissemination*

As TCG attestation tickets are transient by nature, trust information in GTM will soon become obsolete. On the other hand, as nodes are attesting to each other due to occurring events, newly generated attestation tickets can be sent to the set of corresponding nodes to update the trust information maintained in their GTMs:

$$I^a_b = \{N_i \mid \forall i(G^a_{i,b} \neq (0,0))\}$$

I^a_b denotes the set of nodes that have also attested to N_b. As in Figure 3, a new attestation ticket $t_{0,1}$ from N_{a_0} to N_{b_1} is sent to the nodes that have also attested to N_{b_1}: N_{b_2}, N_{b_3}, N_{c_6}, and N_{c_8}. On receiving this ticket, those nodes first verify the signature of the AIK [4] signing the ticket, and then update $t_{0,1}$ in their GTMs.

5. CLOUD TCB ATTESTATION

In this section, we will show that besides achieving fine-grained trust management in the cloud with RepCloud, a higher level of trust semantics can be deduced, which can be used by users to perform cloud TCB attestations. We first propose the concept of Confidence of Trust to represent how much trust towards a node can be inferred from its attestation history, and present the algorithm to calculate it. A preliminary cloud TCB attestation procedure is then presented. We leave the sophisticated definition and analysis on COT together with corresponding attestation architecture and protocols to our future work.

5.1 Confidence of Trust

The *Confidence of Trust (COT)* identifies *the confidence in the node being trustworthy up until a particular time and the confidence for that the trend in the node's configuration will remain trustworthy.* By trustworthy, we refer to the TCG [4] definition as *the capability to genuinely report its own state.* COT of a node is deduced from the following patterns in its *attestation history*: a) the total number of attestations that have been performed towards it in a previous period of time; and b) the mutual-attestation relationship among the nodes that have attested to it in a previous period of time; and c) the freshness of these attestations.

Higher COT denotes lower probability for the node to be corrupted without being identified. Moreover, as in current cloud systems the load of every node tends to be balanced, the change of the COT can be made smooth. Further algorithms can also be proposed to serialize the change of COT, e.g. repeatedly attestation based on adaptive timers or counters. Hence the future attestation patterns towards a node can be approximated, i.e. higher COT also implies that in a smaller time interval the node will be attested to again.

According to the notations in Sec. 4.1, the attestation history of N_j maintained by N_i can be determined from the *jth* column of G_i. As in Figure 3, the attestation history of N_{b_1} contains $\{t_{0,1}, t_{2,1}, t_{3,1}, t_{6,1}, t_{8,1}\}$. COT can be calculated as:

$$C^i_j(cur) = \sum_k \frac{R^i_{i,k}}{F^i_{k,j}(cur)},\ \forall k(ts^i_{k,j} > cur - per)$$

Among the parameters, *cur* is the time of the calculation. k refers to all nodes (N_k) that have attested to N_j within the *per* period of time before the calculation. $R^i_{i,k}$ is the relevance of N_i and N_k. $F^i_{k,j}(cur)$ denotes the freshness of $t^i_{k,j}$. The summation reflects the total number of the attestations toward N_j.

We define the relevance between two nodes as the number of nodes they have both attested to. Higher relevance means the nodes' communication patterns are more common, and hence more common presumptions on trust they may have[25]. It is normalized as the ratio of the number of the nodes in the LTV of N_i to the number of the nodes N_i and N_j have both attested to:

$$R^i_{i,j} = \frac{|\{N_k \mid \forall k(t^i_{i,k} \neq (0,0) \wedge t^i_{j,k} \neq (0,0)\}|}{|L^i|}$$

The freshness of the attestation is calculated as below. An exponential function is used to place more weight on recently performed attestations. *intv* can be specified for different scenarios to normalized the freshness.

$$F^i_{k,j}(cur) = 2^{\frac{cur - ts^i_{k,j}}{intv}}$$

With this definition of freshness, COT can be easily updated by shifting the original value right $(cur' - cur)/intv$ bits, and added the $C^i_j(cur')$, $\forall k(ts^i_{k,j} > max(cur' - per, cur))$, where cur' denotes the new timestamp. This iterative calculation also helps to maintain the history of the attestations performed repeatedly by the same node, while in a GTM every new $t_{i,j}$ overwrites the previous one.

5.2 Preliminary Attestation Procedure

With RepCloud, cloud TCB attestations enable users to determine: a) the core TCBs of the nodes hosting the their VMs are trustworthy (as expected); b) the nodes these nodes depending on are trustworthy; and c) a convincing degree of confidence for this trustworthiness can be deduced. Three steps are considered for performing the remote attestation to the cloud TCB of an application:

Attest to the entrance. As in RepCloud, with decentralized local attestation and trust aggregation and dissemination, nodes are constrained in a web-of-trust that corruption and deception can be effectively identified. In this case, one or more nodes hosting users' VMs can be chosen as the entrances to this web-of-trust. Users or a Trusted Third Party (TTP) may first attest to this entrance, and can hence deduce the trustworthiness of the entire web. The integrity of RepCloud mechanisms are at the same time verified.

Examine the TCB. The LTVs and GTMs of the entrance are then examined to ensure that every node in its TCB is trustworthy within the passed certain period of time. Property-based attestation [12, 9] can further be employed to protect the privacy of the cloud infrastructure, as all the attestation tickets representing these properties can be found in the GTMs.

Evaluate the COT. The COT of the entrance node is used as a reference value for evaluating the COTs of the nodes in its TCB. A more meaningful reference COT value can also be determined according to the user's requirements, the provider's SLA or the TTP's criteria. Pre-trust [13] can also be defined for particular nodes, e.g. Cluster Controller, better to reflect the architecture of the cloud. As trustworthiness of the entrance is just verified, the nodes with a higher COT can also be regarded as trustworthy. On the other hand, as the trust dependency upon the nodes in the TCB is different, weight can be applied to the COT of each node. For the nodes with low COT value, additional local attestation to it can be triggered by the entrance. This newly triggered local attestations can either increase the COT value after success verifications of target nodes, or otherwise report the faulty nodes. As we discussed in Sec. 4.2, after the faulty nodes have been fixed, e.g. re-initialized, they will be assigned with new identities, e.g. new AIKs, and regarded as new nodes. In this case, the COT values for their previous states will be diminished automatically. Evaluation of COT values can also be performed with local attestations to improve the trust dissemination efficiency while preserving a certain degree of attestation frequency for certain nodes.

In this paper, we only consider the trustworthiness of the cloud TCB of an application. In our future work, we will design architecture for attesting to the detailed properties of each node in the cloud TCB of an application, regarding to its own security requirements.

6. EXPERIMENTS

We implemented our RepCloud protocols and simulated the cloud environment for evaluation. We counted the number of actions performed, e.g. total interactions and total attestation, and compared them with the combined scheme and separated scheme (Sec. 1). With performance measurements for those atomic actions in real systems, real-world performance can easily be deduced.

6.1 RepCloud Simulator

The RepCloud simulator is built on top of a P2P simulator, the PeerSim [19]. We use the overlay network in PeerSim to simulate the interactions among nodes in the cloud, e.g. NCs and CLCs. We further implemented a new layer of overlay network for the simulation of VM interactions.

Node Model: We differentiate two kinds of nodes: *managers* and *hosts*, representing CLCs (Cluster Controllers) and NCs (Node Controllers) respectively. SC (Storage Controller) can be implemented as part of the CLC. Security components participate in the communication of cloud applications and are hence regarded as hosts. VMs are grouped into cloud applications, and applications are assigned to hosts in a same cluster with round-robin deployment strategy [2]. Every host can run a number of VMs simultaneously. Each VM runs for a *length* of time, and halts afterwards. A cloud application halts when there is no VM remaining.

The *size* of an application is the number of VMs it contains. We specify a baseline for the size, and a *similarity* value to denote a percentage for the size of each application that can be larger than the baseline, i.e. applications have random sizes within the range from *size* to $size \times (1 + similarity)$. Similarly, we specify the *VM similarity* as the diversity factor for the VMs inside an application. The length of a VM is also specified according to a base value and this factor.

The integrity state of a host is represented as integer numbers. The changes in state can result from malicious behaviors, as we discussed in our adversary model, or from trustworthy operations, i.e. applying security patches, and they can only be discovered by attestations. However, in our experiment, we do not differentiate the trustworthiness of a state, as our main purpose is to evaluate how the changes in state are discovered. We also do not consider the internal states of VMs.

Network Model: A manager connects to a number of hosts to form a cluster. It communicates with them for management tasks, e.g. load-balancing instructions. Managers of the clusters are connected to a root manager to form the cloud, which is regarded as the CC (Cloud Controller). A VM only communicates to the VMs that belong to the same application as its, and cloud applications usually have different communication patterns. However, the aggregated patterns of a host tends to have a random distribution, especially when considering the multi-tenancy and dynamic nature of a cloud. For simplicity, we hence specify each VMs to randomly communicate to others in the same application. We define the *frequency* of a VM as the number of communication actions it performs to another VM within a cycle. Total communications in an application performed within a cycle hence equal to $size \times frequency \times (size - 1)$. Frequency can also relate to VM similarity, in which case the total number should be calculated separately. Communications among VMs trigger communications among hosts, as they are enforced by the hosts.

Simulation Execution: In the initialization phase, hosts and managers are generated and linked accordingly. Cloud applications are deployed to occupy the full capacity of the cloud, i.e. the total number of VMs that can be run simultaneously. New applications are deployed regularly, to keep the overall load stable. VMs are also migrating among nodes inside a same cluster for simulating the effects of load balancing.

Simulation is executed in cycles, in each of which every host executes all its VMs by fetching a list of target communication VMs and generating communication events (actions) to the hosts of the targets. We use the event and time model of PeerSim: time proceeds with the occurrence of events. Every event is generated with a timestamp, and is executed in sequence. The global time is set to the timestamp of the currently processing event. We define the time for performing a communication action as our basic time unit, which is mapped to a millisecond for better presentation. Complicated communications can hence be regarded as a sequence of actions. With the time mapping defined, our experiment results can be easily mapped for real system analysis. In the following text, we refer to the time as our simulated time by default. In our simulation, different kinds of events use different random number generators, e.g. cloud app deployment, migration or attacks, to keep their indecency and achieve the same VMs interaction pattern.

6.2 Cloud Attestation Schemes

Three attestation schemes are simulated and evaluated, a *centralized scheme* (CEN), a fully *decentralized scheme* without reputation systems (DECEN) and our *RepCloud scheme* (REP). We further modified CEN and DECEN to implement fine-grained cloud attestation for evaluation, i.e. to enable each node to determine the trust state of its own cloud TCB with these two schemes.

The centralized scheme is used in the combined attestation [22, 15]. Attestation is performed from the managers to their connecting hosts regularly. Managers are attested to by higher-level ones iteratively. In existing proposals, VMs on each host do not have the knowledge of the properties of the target host it is communicating to. Users can only attest to the Cloud Controller, the root of the attestation hierarchy, which can then report the properties of all nodes that users' VMs are relying on. However, As we discuss in Section 1, it is too complicated to be practical. In our experiments, in order to achieve fine-grained attestation with this scheme, attestation tickets generated by a CLC are broadcasted to all its hosts. Communications should hence be limited within a cluster.

Table 1: Basic Simulation Parameters

Simulation	Length of a simulation cycle (minutes)	1
	Total simulation time (hours)	12
	Bootstrap time (minutes)	10
	Attestation freshness (seconds)	10
Network	# of clusters (CLCs) in the cloud	3
	# of nodes (NCs) per cluster	100
	# of VMs per NC	20
	# of total simultaneous VMs	6000
Cloud Apps	app generation interval (minutes)	1
	# of app generated per interval	60
	# VMs per app	10
	length per VM (minutes)	10
	VM frequency	300
	App similarity	2
	VM similarity	0.6
Attackers	range of attack intervals (seconds)	1 - 10
	# of host to attack per interval	1

DECEN is used in the separated cloud attestation scheme [23]. Attestation should also be performed among nodes to achieve fine-grained cloud TCB identification and attestation. Otherwise detailed cloud infrastructure should be expose to user for attestation. However, trust dissemination and aggregation are not used. Every node maintains its own view of the state of its neighbours. In RepCloud scheme, before a host is communicating to another, it first searches its LTV and GTM for a valid attestation ticket to the communication target. On search miss, it will perform the attestation, and disseminate the ticket as we discuss in Section 4.

The detailed configuration of our simulations is presented in Table 1. *Attestation freshness* denotes that attestation tickets can only be valid for 10 seconds. We simulated 600 applications running at the same time in the cloud, each of which contains 10 to 30 fully connected VMs. An application runs for 10 to 16 minutes with a new one deployed after it terminates. The node communication patterns are hence changing from time to time. Each VM generates 300 to 480 communication actions per second to every VMs in the same app.

6.3 State-change Detection

We simulate attackers who change the state of a random host in random intervals after the bootstrapping phase. As we do not consider the effects of malicious behaviors in our simulation, trustworthy state-changes are also included. We evaluate RepCloud performance with the state-change detection overhead. Two important criteria are considered: 1) *Total Tampered Interactions:* all communications made towards a host with its presumed state (the state deduced from the latest attestation) different from its current state, as they are relying on trust states that have already changed. 2) *Total Attestations:* the total number of attestations performed for discovering the state changes of nodes.

Figure 4 depicts an incremental statistics of the state change detection overhead of the three attestation schemes. The data is refreshed every 10 minutes. Attestation counts

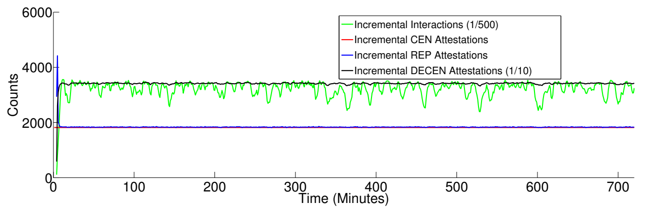


Figure 4: Incremental State Change Overhead

Table 2: Total Tampering Detection Overhead

	Interactions	Attestations	Tampered
REP	$1.15E9$	$1.30E6$	$1.79E6$
CEN	$1.15E9$	$1.30E6$	$1.78E6$
DECEN	$1.15E9$	$2.4E7$	$1.76E6$

of the centralized scheme (CEN) is a strait line, as managers perform attestations to their hosts in a pre-defined interval (10 seconds). In RepCloud, attestation counts boost to a high level in the first few minutes, as it is in the bootstrapping phase and the LTVs and GTMs are mostly empty. Later data shows that RepCloud still exposes stable attestation patterns while achieving only a slightly higher level of average attention counts. For the decentralized scheme (DECEN), as it does not share attestation results among nodes, every node will attest to its neighbour when the tickets are regarded as obsoleted (after 10 seconds). Hence the attestation counts are much larger than the others (around 20 times from our results). Table 2 shows that, RepCloud achieves fine-grained cloud TCB attestation with only a slight increase in attestation overhead, and without reputation systems, redundant attestation efforts will boost significantly (as in DECEN).

6.4 Trust Dissemination

Ticket reporting. In RepCloud, every time when a new attestation ticket towards a target node is generated, it is sent to all the node that have also attested to the target, according to the attestation relationship in the GTM. However, in the centralized scheme, the manager does not possess the communication patterns of each host. Hence, it can only broadcast the new ticket to every node as we discussed before. As we can see from Figure 5, the tick dissemination count is around 6 times as RepCloud's, while the total attestation and interaction counters remain in the same level. Besides, in centralized scheme, VMs of an application can only be deployed within a same cluster to reduce the broadcast domain. However, in RepCloud, an application can scattered among clusters, while remaining a low level of dissemination overhead, as tickets are disseminated according to nodes communication patterns. More importantly, RepCloud reserves the mutual-attestation relationship, from which a higher level of trust semantic (the COT) is deduced.

Moreover, as the communication patterns are changing, communication relationship is also transient. For example, N_a may no longer interact with N_b as all related VMs on it are either halt or migrated out. But $pcr_{a,b}$ may still exist in other nodes' GTM, and new tickets towards N_b are still kept sending to N_a. An obsolete interval can be defined to further reduce the ticket disseminate overhead in Rep-

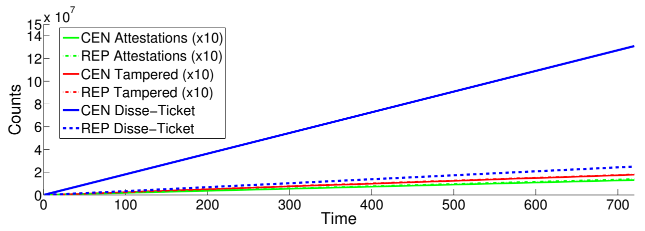


Figure 5: Total Ticket Dissemination Overhead

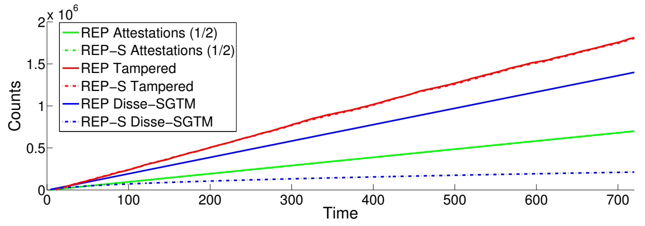


Figure 6: Total SGTM Dissemination Overhead

Cloud. Attestation tickets generated before the interval will not reveal attestation relationship among the nodes.

SGTM fetching. Besides ticket disseminating, RepCloud incurs an extra overhead: SGTM of a target node is fetched every time when a local attestation towards it is performed. However, as newly generated attestation tickets are disseminated to the corresponding nodes, GTM of each node is kept updating continuously. Hence each SGTM fetching may only contain few updates, especially when the communication patterns of nodes are in a relatively stable phase. To reduce the overhead, SGTM fetching can be delayed for an interval of time (1 minute in our case) when the updated entries of the last time is lower than a threshold. We also specify cultivated delay interval, i.e. the interval will be increased (by 1 minute) every time when the both the current and the last update are lower then the threshold. The interval is reset once the update rate is higher than the threshold.

In our experiment, we specify the threshold as whether the size of GTM is changed (i.e. whether new entry is added), as it indicates the changes in communication targets of a node. All the other changes can be updated by ticket dissemination. Figure 6 shows that 86% of the total SGTM dissemination counts are saved (REP-S), while preserving the same level of total attestation and interaction counts. We scale the total attestation counts to half for better presentation, as it overlaps with the original SGTM dissemination counts.

6.5 Security Analysis

As reputation systems are used in RepCloud to disseminate TCG trust, existing attacks to them should be considered [11]. However, we argue that as RepCloud transmit tamper-proof TCG attestation tickets instead of raw reputation values, these attacks are avoided by default. For example, in reputation-based P2P trust management systems, nodes can disseminate forged trust information to promote its own reputation value or to degrade others. In RepCloud, the attestation tickets are signed by the AIK [4] of the platform, the private part of which can only be accessed by the TPM. We will present detailed countermeasurements to these known attacks to reputation systems in our future work.

Malicious cooperative [11] is another kind of attack to reputation systems, with which several tampered nodes cooperate to promote the reputation value of each other. In RepCloud, when all the nodes hosting the VMs of an application are tampered, they may report each other as trustworthy without being identified, and hence tamper the target application. However, with the multi-tenancy nature of the cloud, each of these hosts may at the same time host VMs from other applications, which may communicate to nodes in the cloud TCB of the those application. The cooperative should hence also incorporate these nodes. We argue that the cooperative should be sufficiently large to contain a web-of-trust closure for tampering a target application, e.g. by tampering the entire cluster. On the other hand, in the cloud TCB attestation procedure, an entrance node to the web-of-trust is attested to by a trusted third party. Hence, the larger the malicious cooperative is, the higher probability it will be discovered.

Greedy VM deployment policy [2] can be used in some cloud systems, which exhausts a node before moving on to another node when deploying applications. Hence, VMs from a same application may reside on a same host. As self-attestation (a node to prove its own state to its VMs) is not considered in RepCloud, attestations among these VMs are not performed. However, we argue that it is highly improbable for a node in the cloud to have no communications with others. And any single interaction to others may initiate local attestations as with the adaptive nature of RepCloud. Besides, when all the VMs reside on a same host, the host becomes the entrance node to this application, and will be attested to when cloud TCB attestation is initiated. Moreover, centralized attestation scheme can be incorporated with RepCloud. Managers can participate in the trust dissemination process, and will initiate attestation a node when necessary.

Attacks to the TCG framework have already been concerned in the Timestamp-based attestation scheme [24], which we used as the local attestation in RepCloud. Nonce in the attestation ticket is used to avoid reply attacks, and session keys are used to counteract man-in-the-middle attack. As we discuss in our adversary model, we do not consider runtime attacks to TCG mechanisms. We also do not consider hardware-based attacks.

7. IMPLEMENTATION

In this section, we present the architecture of our RepCloud supporting system and show how it can be implemented with existing trusted computing software stack. We will port our RepCloud protocols from our simulator to it in our future work and evaluate its overhead. From the simulation results we gathered, we envisage that RepCloud will not involve much more overhead than the existing proposals.

Local attestation in RepCloud can be implemented by integrating the trusted computing infrastructure into every node in our cloud model (Sec. 3.1, including the NC, CLC, SC and all other critical hardware resource. RepCloud mechanisms can be added to this software stack and enforced before (GTM searches) and after (trust dissemination and aggregation) the attestation action. The architecture of an NC in our prototype is depicted in Figure 7. Other nodes have the similar architecture except the virtualization layer.

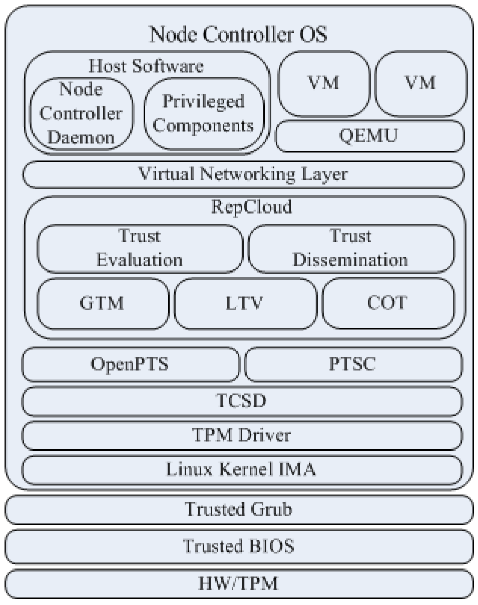


Figure 7: RepCloud Architecture

From the bottom, the Trusted BIOS initializes the TPM at the very beginning of the platform bootstrapping procedure. It measures and loads the trusted bootloader [5], the Trusted Grub in our case, which then measures and loads the Linux Kernel. The IBM Integrity Measurement Architecture (IMA) [21] is enabled in the kernel, which will measure all the critical components, including user space programs, their configuration files, and kernel modules before loading them. A trust chain is hence constructed from the CRTM (Core Root of Trust for Measurement), which resides in the Trusted BIOS and is trusted by default, to the virtualization layer. The TPM driver and the TCSD (Trusted Core Service Daemon) [3] expose Trusted Computing Services to applications. In this paper, we do not consider attestation to VMs, which can be achieved integrating the virtual TPM facilities in our architecture and deploying the trusted infrastructure inside each VM.

Local attestations are implemented with the OpenPTS [7] facilities. The PTSC module manages the trust information of the local platform. It collects the trust evidence for the trust chain and builds a model reflecting the current state of the platform. All the modules in Figure 7 except the VMs compose the core TCB of the node. Every executable in it is measured before loaded, and all related configuration files and data structures, e.g. the GTM, are protected from unauthorized access. This protection mechanism (e.g. access control module) and its policies are also measured.

As we assume adversaries to have full control over a node, the core TCB can still be changed. However, the tamper-proof nature of TPM guarantees that all the changes are genuinely measured. More sophisticated mechanisms for protecting the runtime states of a platform, e.g. DRTM (Dynamic Root of Trust for Measurement) [18], can be used in our system with minor changes to this architecture.

The OpenPTS module validates the target platform by remote attestation. It fetches and verifies the evidence and the model generated from the PTSC on the target, and then translates the model to security properties. As we disseminate and aggregate TCG attestation tickets among nodes, property-based [9, 12] local attestation can be easily achieved. We leave the property-based COT deduction and cloud TCB attestation in our future work. Other OpenPTS facilities are not depicted in this figure.

As RepCloud identifies the cloud TCB of a node from its communication history (Sec. 3.3), all its inbound and outbound traffic should be monitored. RepCloud components are hence implemented under the Virtual Network Layer of the NC, which has full control over all networking traffic. RepCloud filters the traffic, identifies the target node and invokes the Trust Evaluation (TE) module to determine its trustworthiness. TE searches the LTV and GTM for trust evidence and evaluate the COT. It invokes the OpenPTS module to perform local attestation when the evidence is obsolete. Trust Dissemination (TD) module implements the core RepCloud protocol for aggregating global trust. A node is identified with its AIK certificate [4], which is bound to its TPM, to avoid impersonation. PKI facilities are used for managing AIKs and all other certificates. Infrastructure in [24] is also employed for managing the nonce certificates and synchronize the time for all nodes.

8. DISCUSSION

Reputation vs. TCG trust. The meaning of trust in RepCloud is different from trust in Peer-to-Peer systems or Social Network Service system, hence the purposes for incorporating the reputation systems are also different. Trust in those systems only represents a “personal” view of a node, while trust generated by the TCG framework is both deterministic and tamper-proof, i.e. one attestation tickets from a single node can represent the genuine state of the target node, and this statement cannot be forged or modified. Moreover, the evaluation of trust in reputation systems can only be performed after the actual action has been taken place, but may have a relative long endurance, i.e. the more trustworthy a node (a person) is, the less possible it will lie. Meanwhile, TCG trust can be obtained before the action, but every node has the same probability of being corrupted in the next period of time. Hence the main purpose of RepCloud is to reduce unnecessary attestation overheads while preserving a low interval for repeating attestation. Still, as they share the similar trust dissemination and aggregation patterns, algorithms in reputation systems can still be adapted in RepCloud.

Trust represented as probability. In this paper, we propose to represent the trustworthiness of the TCB of a cloud application as a degree of probability – the confidence of trust (COT). This may seem violating the philosophy of the TCG trust for being deterministic and tamper-proof. However, we argue that, as with the discrete nature of TCG attestation, many TCG-compliant applications, including the current trusted cloud solutions, are still implicitly replying on the probability that the configuration of a target platform will not change between the time-of-attest and time-of-interact. We argue that as with the web-of-trust constructed by trust dissemination in RepCloud, COT values vouch for a stronger guarantee for trustworthiness than existing attestation schemes.

9. RELATED WORK

Trusted Virtual Datacenter (TVDc) [8] incorporates trusted computing technologies into virtualization and system management software. It provides strong isolation between workloads by enforcing a Mandatory Access Control (MAC) policy throughout a datacenter. TVDc also provides integrity guarantees to each workload by leveraging a hardware root of trust in each platform to determine the identity and integrity of every piece of software running on a platform. In addition, TVDc allows centralized management of the underlying isolation and integrity infrastructure.

Trusted Cloud Computing Platform (TCCP) [22] enables users to attest to the IaaS provider and determine whether or not the service is secure before they launch their virtual machines. A Trusted Coordinator (TC) maintained by an External Trusted Entity attests to the nodes inside the cloud, and control the critical operations of the nodes with a set of protocols, such as VM migration and instantiation. Private Virtual Infrastructure (PVI) [15] is a management and security model for cloud computing. The PVI datacenter is under control of the information owner while the cloud fabric is under control of the service provider. A cloud Locator Bot pre-measures the cloud for security properties, securely provisions the datacenter in the cloud, and provides situational awareness through continuous monitoring of the cloud security.

These trusted cloud solutions above share the similar centralized structure and maintain and prove the trust state of the entire cloud altogether. As we presented in Section 1, deficiencies limit the scalability of the cloud. RepCloud implements a decentralized peer-to-peer attestation structure and expose the trust state of the cloud taking considerations of the peer attestation relationships. Confidence of trust is deduced to determine continuous trust from discrete TCG attestation tickets. The semantics of communications are preserved, which can enable a fine-grained attestation, with the security concerned nodes been regularly attested to. With the support for trust aggregation and dissemination, the overall attestation overheads are reduced while still achieving a low level state-change-discovery delay.

Cloud verifier (CV) [23] service generates integrity proofs for customers to verify the integrity and access control enforcement abilities of the cloud platform that protect the integrity of customer’s application VMs in IaaS cloud. Cloud customers can verify that the cloud verifier satisfies their integrity property requirements and that the properties the cloud verifier vouches for being enforced on its hosts satisfies the customer’s properties for those components as well. However, CV can only verify the trustworthiness of the nodes hosting the user VMs, while RepCloud identifies and dynamically manages the effective cloud TCB for the VMs and enables attestations with a higher level of confidence for trust.

10. CONCLUSION AND FUTURE WORK

In this paper we presented RepCloud to explore a new way for managing trust in the cloud by taking advantage the research in reputation systems. We further proposed the concept of the Confidence of Trust to deduce continuous trust from the discrete TCG attestation tickets aggregated in RepCloud, with which we showed that fine-grained and scalable cloud TCB attestation can be achieved. We implemented RepCloud in our simulated cloud environment, and the results showed that, besides achieving fine-grained attestation, RepCloud still enforced the same level of state-change-discovery delay as the combined cloud attestation schemes. Further analysis showed that RepCloud incurred much less overhead than the modified combined and separated attestation for achieving fine-grained cloud TCB attestations. We will propose sophisticated COT calculation and evaluation algorithms and design detailed cloud TCB attestation architecture in our future work. We will also port our RepCloud protocol to the supporting system and evaluate its performance.

11. ACKNOWLEDGEMENT

We especially thank Cornelius Namiluko, Wattana Viriyasitavat, Dr. John Lyle, Dr. Liang Gu and Dr. Qingni Shen for their valuable advice in the research. We would also like to thank the anonymous reviewers for their very valuable comments and their helpful suggestions.

12. REFERENCES

[1] Cloud security alliance. `http://www.cloudsecurityalliance.org`.

[2] Eucalyptus. `http://www.eucalyptus.com`.

[3] Trousers - the open-source tcg software stack. `http://trousers.sourceforge.net/`.

[4] Trusted computing group. `http://www.trustedcomputinggroup.org`.

[5] Trusted grub. `http://trousers.sourceforge.net/grub.html`.

[6] Amazon cloud architecture. `http://jineshvaria.s3.amazonaws.com/public/cloudarchitectures-varia.pdf`, 2008.

[7] Open platform trusted service user's guide. `http://iij.dl.sourceforge.jp/openpts/51879/userguide-0.2.4.pdf`, 2011.

[8] Berger, S., Cáceres, R., Pendarakis, D., Sailer, R., Valdez, E., Perez, R., Schildhauer, W., and Srinivasan, D. Tvdc: managing security in the trusted virtual datacenter. *SIGOPS Oper. Syst. Rev. 42* (January 2008).

[9] Chen, L., Löhr, H., Manulis, M., and Sadeghi, A.-R. Property-based attestation without a trusted third party. In *Proceedings of the 11th international conference on Information Security* (Berlin, Heidelberg, 2008), ISC '08, Springer-Verlag, pp. 31–46.

[10] Christodorescu, M., Sailer, R., Schales, D. L., Sgandurra, D., and Zamboni, D. Cloud security is not (just) virtualization security: a short paper. In *Proceedings of the 2009 ACM workshop on Cloud computing security* (New York, NY, USA, 2009), CCSW '09, ACM, pp. 97–102.

[11] Hoffman, K., Zage, D., and Nita-Rotaru, C. A survey of attack and defense techniques for reputation systems. *ACM Comput. Surv. 42* (December 2009).

[12] Jonathan, P., Matthias, S., Els, Van, H., and Michael, W. Property attestation – scalable and privacy-friendly security assessment of peer computers. In *Technical Report RZ 3548* (2004), IBM Research.

[13] Kamvar, S. D., Schlosser, M. T., and Garcia-Molina, H. The eigentrust algorithm for reputation management in p2p networks. In *Proceedings of the 12th international conference on World Wide Web* (New York, NY, USA, 2003), WWW '03, ACM.

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

[15] Krautheim, F. J. Private virtual infrastructure for cloud computing. In *Proceedings of the 2009 conference on Hot topics in cloud computing* (Berkeley, CA, USA, 2009), HotCloud'09, USENIX Association.

[16] Lyle, J., and Martin, A. Trusted computing and provenance: better together. In *Proceedings of the 2nd conference on Theory and practice of provenance* (Berkeley, CA, USA, 2010), TAPP'10, USENIX Association, pp. 1–1.

[17] McCune, J. M. Turtles all the way down: research challenges in user-based attestation. In *Proceedings of the 2nd workshop on Recent advances on intrusiton-tolerant systems* (New York, NY, USA, 2008), WRAITS '08, ACM, pp. 2:1–2:1.

[18] McCune, J. M., Li, Y., Qu, N., Zhou, Z., Datta, A., Gligor, V., and Perrig, A. Trustvisor: Efficient tcb reduction and attestation. In *Proceedings of the 2010 IEEE Symposium on Security and Privacy* (Washington, DC, USA, 2010), SP '10, IEEE Computer Society, pp. 143–158.

[19] Montresor, A., and Jelasity, M. Peersim: A scalable p2p simulator. In *Peer-to-Peer Computing, 2009. P2P '09. IEEE Ninth International Conference on* (sept. 2009), pp. 99 –100.

[20] Ristenpart, T., Tromer, E., Shacham, H., and 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* (New York, NY, USA, 2009), CCS '09, ACM, pp. 199–212.

[21] Sailer, R., Zhang, X., Jaeger, T., and van Doorn, L. Design and implementation of a tcg-based integrity measurement architecture. In *Proceedings of the 13th conference on USENIX Security Symposium - Volume 13* (Berkeley, CA, USA, 2004), SSYM'04, USENIX Association, pp. 16–16.

[22] Santos, N., Gummadi, K. P., and Rodrigues, R. Towards trusted cloud computing. In *Proceedings of the 2009 conference on Hot topics in cloud computing* (Berkeley, CA, USA, 2009), HotCloud'09, USENIX Association.

[23] Schiffman, J., Moyer, T., Vijayakumar, H., Jaeger, T., and McDaniel, P. Seeding clouds with trust anchors. In *Proceedings of the 2010 ACM workshop on Cloud computing security workshop* (New York, NY, USA, 2010), CCSW '10, ACM.

[24] Stumpf, F., Fuchs, A., Katzenbeisser, S., and Eckert, C. Improving the scalability of platform attestation. In *Proceedings of the 3rd ACM workshop on Scalable trusted computing* (New York, NY, USA, 2008), STC '08, ACM.

[25] Walsh, K., and Sirer, E. G. Experience with an object reputation system for peer-to-peer filesharing. In *Proceedings of the 3rd conference on Networked Systems Design & Implementation - Volume 3* (Berkeley, CA, USA, 2006), NSDI'06, USENIX Association, pp. 1–1.