Secure Attestation of Virtualized Environments

Michael Eckel[1], Andreas Fuchs[1], Jürgen Repp[1], and Markus Springer[2]

[1] Fraunhofer SIT, Rheinstraße 75, 64295 Darmstadt, Germany,
firstname.lastname@sit.fraunhofer.de
[2] Darmstadt, Germany, springerm.de@gmail.com

Abstract. Securing the integrity of virtualized environments like clouds is challenging yet feasible. Operators have discovered the advantages of virtualization technology in terms of flexibility, scalability, cost-effectiveness, and availability. Applications range from network and embedded devices to big data centers and cloud computing. Trusted Computing technology can be employed to protect the integrity of a system by leveraging a Trusted Platform Module (TPM) and remote attestation.
Existing research on remote attestation of virtualized environments differs in scalability, resource consumption, and provided security guarantees. While some approaches scale at large and use the TPM efficiently, they are way more intrusive, requiring changes to hypervisor and Virtual Machines (VMs). Others render entirely impractical with an increasing number of VMs, caused by the TPM being the bottleneck.
In this paper we analyze existing work on remote attestation for virtualized environments and discuss benefits as well as shortcomings. We identify an approach that provides adequate security and is easy to implement but is prone to relay attacks. We improve that approach by developing countermeasures, while maintaining existing security guarantees. Our contribution requires only minimal changes to the hypervisor system, keeping existing attestation protocols intact. We implement and evaluate on production-grade hardware, and compare our improved attestation approach with the most sophisticated alternative approach.
With performance measurements and further evaluations we show that our solution outperforms the other approach for a small number of VMs, as used in network devices and embedded systems.

Keywords: Virtualization · Remote Attestation · Trusted Computing

1 Introduction

Virtualization technology has paved the way for businesses to reduce infrastructure costs while also improving dependability and scalability. Being able to dynamically launch and tear down new VMs on demand, infrastructure costs are cut even further. Especially when handling huge amounts of data on the go, cloud services come in very handy. Based on virtualization technologies, they can be hosted internally or externally, and be migrated back and forth. Depending on the use case, data and workloads handled by VMs may be of sensitive or confidential

nature. Those can be protected against disclosure and manipulation by providing encryption and authenticated channels during transport. However, data can still leak if VMs or hypervisor hosts are manipulated.

To mitigate these threats, Trusted Computing technologies, such as the TPM and remote attestation, can be leveraged. Remote attestation is the process of a verifier making claims about the integrity of target systems, based on executed software. Hypervisors equipped with a TPM allow for this kind of remote attestation. Before data is to be processed on a VM, its trustworthiness could be verified using remote attestation.

Current technologies allow for both attestation of boot time and attestation of runtime. Boot time attestation encompasses all components that are executed in strictly sequential order during system boot, i. e. up to the kernel of an operating system (OS). From there, the runtime of a system begins, executing software applications in parallel in an unpredictable, non-deterministic order. A major problem is that software inside VMs is per se not considered in a remote attestation process. To make sure that VMs behave as expected, tools like Intel's Open Cloud Integrity Technology (Open CIT) implement features to measure the integrity of VM images prior to their execution. This approach turns out to be very time-consuming and inflexible since it supports only static images.

Hypervisor solutions like KVM support passing through the Physical TPM (pTPM) to VMs. That allows for identical attestation technologies to be used for VMs and hypervisor. However, forwarding the same pTPM to many VMs, makes the TPM a bottleneck and introduces race conditions. Virtual TPMs (vTPMs) are software implementations of a TPM which can be assigned to VMs, conveying the impression to a VM of using a pTPM [2,3]. This way the same attestation technologies as for physical machines can be used. However, integrity of VMs depends on the integrity of the underlying hypervisor.

In this paper we analyze existing attestation approaches for virtualized environments. We improve the existing *separate attestation* approach to strengthen it against relay attacks. We implement the hypervisor-based attestation approach as proposed by Lauer and Kuntze [12] as well as our improved separate attestation approach. Eventually, we evaluate the feasibility of both approaches with performance measurements on production-grade hardware.

In Section 2 we analyze related work on attestation for virtualized environments. We present our target reference architecture in Section 3. Section 4 identifies attestation requirements which we consider vital. Section 5 then presents our main contribution, the strengthening of the separate attestation approach. In Section 6 we describe our implementation, followed by our evaluation in Section 7. Section 8 summarizes the paper and finishes with potential future work.

2 Related Work

Existing research on attesting virtualized environments focuses on either providing isolated attestations of the hypervisor or of single VMs [11,17]. Some propose

more sophisticated solutions that attempt to build a trusted virtual machine monitor (VMM).

One of these approaches is Terra from Garfinkel et al. [10]. Terra protects VMs by isolating them from each other. It measures individual blocks of memory prior to loading them and enables remote attestation to external appraisers. This memory introspection approach is computing-intensive and requires a huge amount of main memory. However, Terra builds a basis for many other work which further elaborates on the idea of protecting VM memory, such as Cerberus [6] and TrustVisor [13].

Other approaches, such as the Hypervisor-Based Integrity Measurement Agent (HIMA) by Azab et al. [1], measure programs and services running inside a VM. Prior to execution by the hypervisor, VM system calls and hardware interrupts are intercepted and analyzed. It requires NX bit (no-execute bit) support on the hypervisor to mark measured memory pages prior to their execution. HIMA requires all memory pages of a program to be loaded into memory. As a consequence, changes to the OS are necessary. HIMA requires a large amount of computing power and its applicability must be carefully evaluated.

There exist CPU based attestation approaches to achieve software integrity protection, such as Bastion [5] and HyperWall [15]. These CPU extensions allow for measuring and verifying software modules prior to their execution and were analyzed by Zhang et al. in [18] regarding their applicability to cloud solutions.

Since memory verification techniques based on TPMs are slow, there was a desire to supply VMs with TPM functionality in a resource efficient manner. The work in [14] introduced an abstraction layer inside the VMM, adding context switching functionality in a pTPM. This provides VMs with the illusion of having exclusive access to a real pTPM. The problem with this approach is that sharing one physical TPM with many VMs makes the pTPM a bottleneck. With the introduction of vTPMs by Berger et al. [2], attestation of VMs changed to some extend. With vTPMs it is possible to supply every VM with its own TPM in software which is securely anchored in a pTPM. Since its introduction in 2006, Berger et al. have altered the vTPM implementation slightly, turning it into a hypervisor extension. Thus, eliminating the need for an additional management VM [3]. Due to its efficiency, many researchers now focus on the integration of vTPMs to attest software running inside VMs. Migration of VMs and associated vTPMs is another topic that is actively researched [17].

While the majority of existing research treats attestation of VMs and hypervisors separately, almost no research on binding these attestations together is conducted. A verifier may trust a VM after verifying its integrity, using remote attestation. However, if the underlying hypervisor is compromised, the integrity and trustworthiness of the VM may also be compromised. In case an attacker manages to compromise a hypervisor, she may be able to extract or manipulate sensitive information, e. g. by tampering with VM main memory.

The Virtualized Platform working group of the Trusted Computing Group (TCG) defined a deep attestation scheme [16] to link attestations of VMs and

hypervisor. In [4], Celesti et al. describe how remote and deep attestation can be used to build federated cloud environments.

Lauer and Kuntze propose Hypervisor-based Attestation (HYPA) [12]. It adds an Attestation Manager (ATAMAN) to the hypervisor which directly reads the internal state of running vTPM instances. The approach falls into the category of bottom-up attestation approaches and aims at improving scalability of VM attestations, in cases where tens or hundreds of VMs are running.

In a HYPA process, the Verifier sends a single attestation request to the hypervisor (cf. Figure 1, 1). In return, it receives a single attestation response (9), including attestations of all running VMs (2–7) as well as the hypervisor itself (8).

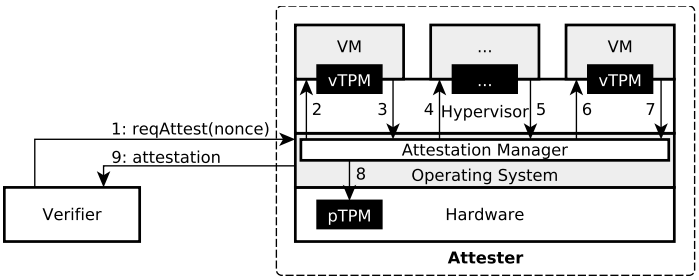

Fig. 1: Hypervisor-Based Attestation (based on [12])

The Attestation Manager directly accesses running vTPM instances to read their internal state, i. e. vPCRs. Furthermore, it reads the *virtual* Stored Measurement Log (vSML) from inside each VM. All vPCRs and vSMLs are compiled into a data structure, which is then hashed. The resulting digest is passed as qualifying data (nonce) to a *TPM Quote* operation of a hypervisor attestation, using the pTPM. The Attestation Manager runs within the bounds of the hypervisor, and as such, it is part of the hypervisor attestation. The overall attestation result includes the data structure with the vPCRs and vSMLs as as well as the hypervisor attestation. The Verifier verifies both the hypervisor attestation and the VM attestations.

The main benefit of the hypervisor-based attestation approach is its huge scalability improvement compared to individual VM attestations. This is reflected by the reduction of attestation protocol overhead as well as reduced I/O operations. However, it is very intrusive, requiring access to currently running vTPM instances as well as access to VM internals, i. e. the vSMLs.

Separate Attestation (SEPA) of VMs and hypervisor is another approach described by Lauer and Kuntze in [12]. The Verifier sends an attestation request to a single VM (cf. Figure 2, 1) and receives an attestation result (2). Then the Verifier sends an attestation request to the hypervisor (3), using the same nonce, and receives the attestation result (4). By using the same nonce, a weak layer linking is achieved between the VM and hypervisor.

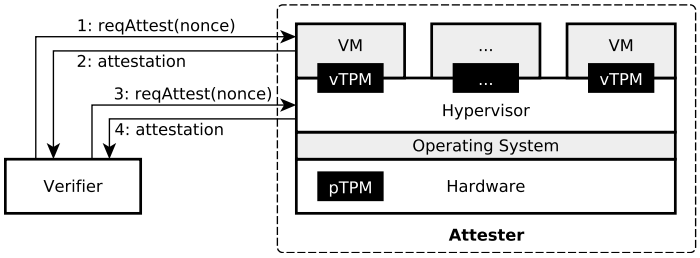

Fig. 2: Separate Attestation (based on [12])

As stated by the authors, the SEPA approach is not resource-conserving and makes the pTPM the bottleneck during consecutive VM attestations. The fact that each individual VM attestation also requires an additional hypervisor attestation, reduces the scalability of this approach drastically. Another problem is that layer linking is not very strong and a Verifier requires constant up-to-date information on which hypervisor a VM is running. Since this kind of layer linking is neither enforced nor verifiable, it undermines the strong attestation statements of pTPM and vTPM. Lauer and Kuntze propose a way to improve the linking between VM and hypervisor by providing the VM with a secret value N which is used to generate remote attestation data. After the VM terminates, it provides the hypervisor with the secret value N over a secure channel between VM and hypervisor. The hypervisor then uses this secret value for attestation. The downside of this approach is that the Verifier does not treat VM and hypervisor equally. It requires changes to the guest OS and hypervisor. Thus, complicating the implementation and reducing scalability [12].

SEPA does not require changes to existing software components or protocols, and thus can be easily integrated into existing attestation solutions. It is possible to treat VMs and hypervisors equally from a Verifier point of view. In systems which only run a small number of VMs, this approach represents a feasible attestation solution. However, we observed that separate attestation as proposed by Lauer and Kuntze is subject to relay attacks.

There exist other cloud integrity verification techniques that are more complex but rely on previously described technologies. In the upcoming section we introduce our reference architecture, which we use to further evaluate the SEPA and HYPA approaches.

3 Reference System

Our reference architecture is based on the one presented by Intel and Hytrust in [8] which splits data into work packages to process them in parallel (cf. Figure 3). We propose to use VMs for that purpose which can be flexibly spawned and teared down on current demands. The Workflow Orchestration handles distribution and scheduling of work packages to VMs. We assume host systems and VMs to be owned and operated by the same party.

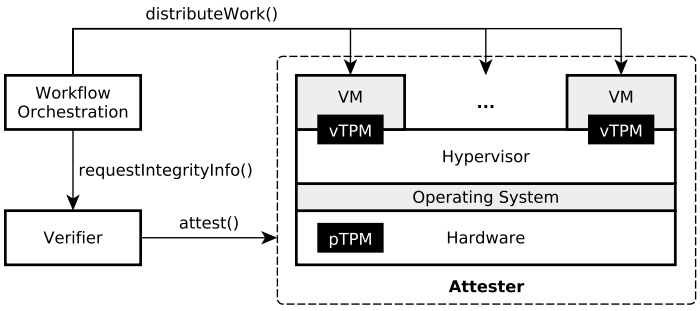

Fig. 3: vTPM-based reference architecture. The Workflow Orchestration relies on the Verifier to verify the integrity of VMs and their hypervisor.

Before entrusting a VM with processing of potentially sensitive work packages, the Workflow Orchestration ensures the VM to be trustworthy. This process is complex and requires in-depth knowledge about the configuration of a particular VM. That is why a Verifier is dedicated to that task. It attests hosts and VMs to provide the Workflow Orchestration with up-to-date integrity information. The Verifier maintains a database of Reference Integrity Measurements (RIMs) which represent known good system configurations. By comparing them to the current operational state of a system, the Verifier decides whether or not hosts and VMs to be trustworthy.

Hosts are equipped with a pTPM and run Linux. They have a TPM-enabled BIOS and boot loader, and support Measured Boot for boot time integrity, and the Linux Integrity Measurement Architecture (IMA) for runtime integrity. The VM hypervisor supports vTPMs to be assigned to VM, such as QEMU/KVM. vTPMs are executed in the context of the hypervisor and one vTPM is assigned to exactly one VM [3]. vTPMs provide identical functionality as pTPMs. However, they lack a True Random Number Generator (TRNG) as well as an Endorsement Key Certificate (EK Cert) from the TPM manufacturer. Further, they cannot provide the same security guarantees as a pTPM as they are only software.

4 Attestation Requirements

Coker et al. [7] already postulate requirements for remote attestation based solutions. Lauer and Kuntze analyzed and extended them in [12]. The first of these requirements is the freshness of attestation information, which requires to keep the gap between the time of a measurement and the time of its use as small as possible to minimize the risk of using outdated information. The next requirement is the comprehensiveness of attestations to enable the verifier to deduct the state of a system in a replicable, accountable and comprehensive manner. The attestation information must also be presented in a logical form that helps the verifier to correlate multiple attestations over time to further improve the detection of misbehavior. Since attestations expose sensitive information

about a target system, their disclosure must be constrained to valid verifiers. In order to build trust into an attestation, the verifier relies on the trustworthiness of delivery mechanisms which ensure authenticity and integrity of reported data.

In terms of cloud attestations, Lauer and Kuntze further require layer linking between VMs and hypervisor in order to trust VMs attestations [12]. Hence, there must be valid attestation information available for both VM and hypervisor. And it must be possible to correlate these in order to ensure a VM runs on its assigned hypervisor. Another requirement by Lauer and Kuntze is scalability of the attestation scheme, which targets systems with a large number of VMs. In terms of TPM-based attestations, that means to reduce the number of calls to the pTPM [12].

In addition, we propose implementation-specific requirements: *implementation complexity* and *guest independence*.

Implementation Complexity Depending on the used attestation scheme and how attestation information is acquired, modifications to guest OS or vTPM are required, which increase development costs. This also applies to installation of additional software inside a guest system, even though these are not considered as critical as OS code changes. In a typical cloud environment, customers want to chose software and OS based on application needs, rather than whether an implementation for additional security tools exists. Impact and intrusiveness of these tools must be taken into account. The smaller and less intrusive tools are the better they can be ported. Huge code changes and dependencies to other software may yield a bigger obstacle in porting. Required system privileges for the software must also be taken into account.

Guest Independence In order to reduce the attack surface, the attestation solution must rely on as little additional infrastructure as possible. Dependencies to additional infrastructure also influence the overall reliability of the solution. Increasing implementation complexity, results in systems becoming more error-prone. Less complex solutions are easier to maintain.

5 Secure Separate Attestation

During our analysis of the SEPA approach [12], we discovered a flaw in the layer linking mechanism between VM and hypervisor. An attacker could launch a relay attack to generate a verifiable VM attestation and thus fool the verifier into trusting a *Bad VM*.

We assume a threat model following the definitions of Dolev and Yao in [9]. The attacker has full control over the network, and thus can read, modify, insert, or block messages anywhere on the network. However, she is not able to break any underlying cryptographic primitives. Furthermore, the attacker is able to gain control over the system running inside a VM, but not the hypervisor.

To launch her attack the attacker operates her own hypervisor host (*Attacker-Controlled System*) outside the bounds of the target system, as depicted in Figure 4. On that system she runs a pristine copy of the VM alongside with the corresponding vTPM, including its configuration and keys. She further gains

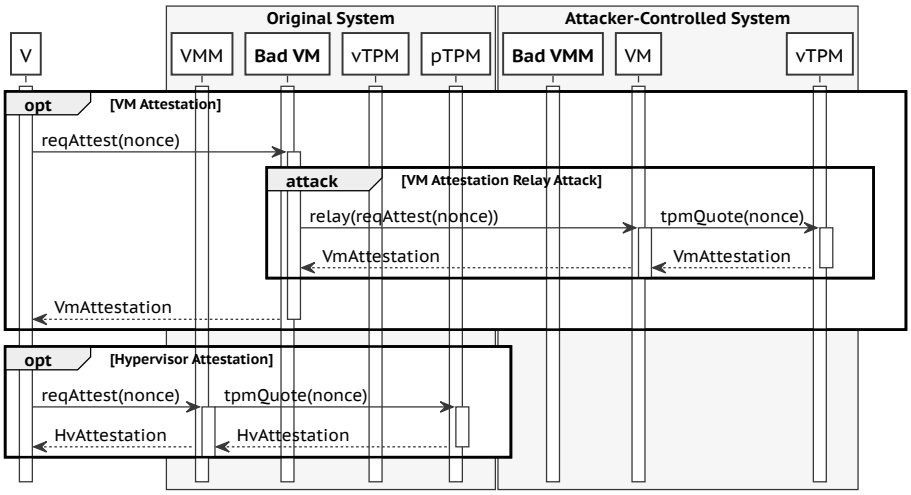

Fig. 4: Relay Attack on Separate Attestation

access to a VM on the *Original System*, in such a way that she can modify and run compromised software inside (cf. *Bad VM* in Figure 4).

Figure 4 shows a time sequence diagram of the relay attack process. Whenever the Verifier (V) sends its attestation request to the VM, the *Bad VM* relays this request to the pristine VM on the Attacker-Controlled System. From there, the VM forwards the request to its vTPM which generates a *VmAttestation*. The VM returns this response to the Bad VM, which eventually forwards the VmAttestation to the Verifier. In a subsequent attestation of the hypervisor on the *Original System*, the pTPM is used. The Verifier in turn verifies both attestations, incorrectly considering the Bad VM trustworthy.

We solve the relay problem by using the vTPM itself as a trusted channel to the hypervisor. Whenever a VM receives an attestation request from the Verifier (V), it triggers a *TPM Quote* operation on the vTPM (cf. Figure 5), as usual. At this point, as a matter of mitigation, we let the vTPM store a hash of the *vTPM Quote* result ($\text{VmAttestation}_{\text{hash}}$) on the hypervisor (3). Then, the usual flow continues, and the VM returns the VM attestation result to the Verifier.

Subsequently, the Verifier sends an attestation request to the hypervisor. The hypervisor uses the most recent $\text{VmAttestation}_{\text{hash}}$ for the VM and includes it in the hypervisor attestation. For that purpose, the hypervisor creates a compound nonce by hashing the concatenation of the nonce from the Verifier with the $\text{VmAttestation}_{\text{hash}}$. This compound nonce is used in the *TPM Quote* operation on the pTPM. The result is returned to the Verifier. The Verifier verifies that the VM attestation hashes match. For that purpose, it creates a hash of the VM attestation data (VmAttestation) it received from the VM attestation. The Verifier generates the nonce exactly as the hypervisor did, and compares it with the compound nonce from the hypervisor attestation data. Only if the compound

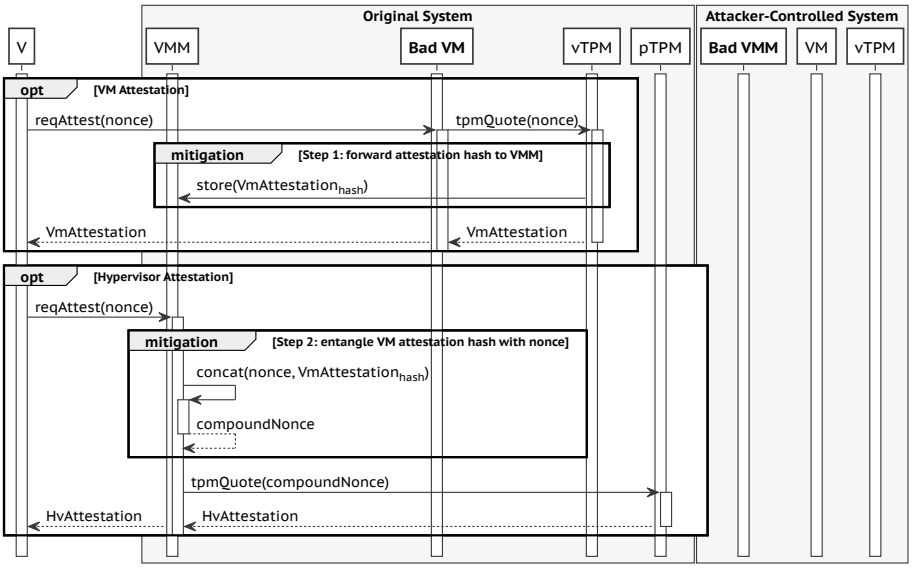

Fig. 5: Secure Separate Attestation

nonces match, the Verifier can be certain it received the correct VM attestation. This way, layer linking between VM and hypervisor is significantly improved. After that, the Verifier continues, verifying VM and hypervisor attestations.

This approach introduces the risk of disclosing the nonce designated to a VM to other VMs or unauthorized third parties. We consider this risk to be minimal, as the attestation approach ensures that both VM and the underlying hypervisor host are trustworthy. Other security measures like encryption and access control mechanisms can be used to further secure the forwarded hashes. A hypervisor that is not trusted could potentially disclose forwarded information from the vTPM anyway, but would be detected using the SEPA scheme.

6 Implementation

Memory introspection approaches are computing-intensive, require a lot of main memory, and are not easy to implement. Intercepting system calls from VMs to the host OS requires changes to the hypervisor as well as the host OS. Because these violate many of our requirements, we do not consider them as candidates for our implementation. Layer-linking is vital and we choose SEPA and HYPA for our implementation, both of which are in accordance with our requirements.

We implement SEPA and HYPA as a single proof-of-concept, extending Intel's Open CIT 2.2. Open CIT features boot time attestation of host machines and uses Intel Trusted Execution Technology (TXT) to establish a hardware Root of Trust for Measurement (RTM). It supports several hypervisors, such as VMware

ESX, Citrix Xen, Kernel-based Virtual Machine (KVM), and Microsoft HyperV. Attestation information is presented via a web interface.

Open CIT consists of two main components: Attestation Server and Trust Agent. The Attestation Server acts as Verifier for remote attestations. It maintains a database with RIMs and attestation results, and exposes those via REST API to relying parties. Trust Agents run on any attested machine handling attestation requests from the Attestation Server. That is, performing *TPM Quote* operations, collecting Stored Measurement Logs (SMLs), and returning those.

We add support for attesting VMs to Open CIT, including HYPA and our improved SEPA layer binding mechanisms to associate VMs with their underlying hypervisor. This requires changes to both Trust Agent and Attestation Server. We develop a vTPM instance manager to provide VMs with vTPMs, using QEMU and Stefan Berger's vTPM implementation, adapted to our needs. That is, for HYPA we add Platform Configuration Register (PCR) read access to the hypervisor Trust Agent, and for SEPA we store VM attestation hashes in the hypervisor file system. For HYPA to access vSMLs inside VMs, we use *VirtIO* on the hypervisor to share a *tempfs* folder with VMs. Further, we add support for runtime integrity verification with IMA. This requires changes to Attestation Server, Trust Agents, and VM OS.

7 Evaluation

In this section we evaluate both SEPA and HYPA attestation approaches. The evaluation system is a Huawei RH5885 V2 server equipped with four Intel® Xeon® E7-8870 CPUs at 2.40 GHz, 10 cores per CPU and 2 threads per core. It provides 1 TiB of main memory and a plug-in physical TPM 2.0. It runs the customized Open CIT Attestation Server in a VM as well as ten distinct client VMs. The hypervisor host uses libvirt, KVM, and QEMU 2.6 for virtualization. All machines run Ubuntu 16.04. Client VMs and hypervisor run the customized Open CIT Trust Agents.

In order to compare performance measurements between SEPA and HYPA, we define evaluation scenarios based on typical actions which Open CIT performs to attest VMs and hypervisors: 1. Attest all VMs, 2. Attest a single VM, 3. Attest hypervisor only.

7.1 Evaluation Criteria

To evaluate our proof-of-concept implementation we define evaluation criteria. The following paragraphs explain them.

Execution Time In order to compare the attestation schemes based on execution time, we measure the overall execution time of each scenario. This includes attestation request generation, response handling, verification, and transport over the network. Since Open CIT produces a lot of overhead in the backend in order to generate and verify the *TPM Quote* response, we also measure the time these two events require to complete.

Table 1: Execution Time in ms

		SEPA			HYPA		
Scenario	**Name**	min	avg	max	min	avg	max
All VMs	Execution time	50097.25	50423.92	50751.25	12907.50	13152.17	13299.00
	Retrieve *TPM Quote*	1494.50	1513.35	1545.25	1594.50	1605.50	1617.00
	Verify *TPM Quote*	15.00	23.31	28.25	21.25	22.92	24.50
One VM	Execution time	4690.50	4790.75	4895.75	12890.50	13091.92	13231.75
	Retrieve *TPM Quote*	1509.25	1518.08	1525.75	1597.75	1610.83	1624.25
	Verify *TPM Quote*	21.25	24.92	28.25	21.25	22.42	23.75
Hypervisor	Execution time	2833.50	2857.00	2890.00	12815.00	13091.17	13258.00
	Retrieve *TPM Quote*	1497.50	1510.17	1521.50	1587.50	1605.17	1623.50
	Verify *TPM Quote*	24.50	25.84	26.50	21.00	21.83	23.00

Network Usage For each evaluation scenario we capture the entire network traffic for three subsequent attestations. For each connection from the Attestation Server to the hypervisor host or a VM, we compute the amount of received and transmitted data, the throughput rate, and the time between the first and last transmission. In our summary, we include only the accumulated amount of data for all connections.

CPU and Memory Usage For three attestations per attempt, we measure the resident (non-swapped) physical memory consumption in megabytes as well as the percentage of CPU utilization. Depending on the scenario, the Attestation Server and the Trust Agent are measured every 100 milliseconds with the Linux *top* command, computing average and peak values. Some tasks consume more than 100 % of the CPU, which means they occupy more than one CPU core.

7.2 Evaluation Results

During evaluation, we observed the expected behavior for the schemes. Table 1 shows measured execution times for the scenarios. We generate ten measurements per scenario and accumulate them into a single table. That is, each value of a group is computed as the arithmetic mean of all measured execution times for a particular scenario.

As shown in Table 1, if all VMs are verified, HYPA is faster than SEPA, as it attests the hypervisor only once. In case only one VM, or only the hypervisor is verified, SEPA is a lot faster. An interesting observation is the time it takes to retrieve a TPM Quote response, and the time to verify it. This seems to be largely unaffected by the amount of attested VMs, with SEPA showing an exceptional maximum peak. In practice, however, HYPA will scale worse if fewer attestations are requested, or if the IMA SMLs become very large in size. After a certain threshold of requested attestations is reached, which depends on the number of VMs, SEPA becomes less efficient than HYPA.

Table 2: Network Usage

		SEPA	HYPA	
Scenario	**Type**	Amount	Amount	Unit
All VMs	Send	0.64	0.06	KiB
	Receive	6317.20	2323.86	KiB
	Time	178.89	63.19	s
	Receive/s	43.11	36.81	KiB/s
One VM	Send	0.14	0.06	KiB
	Receive	592.56	2321.03	KiB
	Time	43.47	62.93	s
	Receive/s	13.64	36.93	KiB/s
Hypervisor	Send	0.06	0.06	KiB
	Receive	378.81	2320.77	KiB
	Time	15.07	62.84	s
	Receive/s	25.15	36.93	KiB/s

Table 3: CPU and Memory Usage

		SEPA		HYPA	
Scenario	**Type**	avg	peak	avg	peak
All VMs	%CPU	50.93	280.50	69.15	217.18
	RES/MB	966.48	1015.85	1045.48	1074.50
One VM	%CPU	58.82	202.43	70.20	225.63
	RES/MB	1146.65	1209.35	1055.06	1091.91
Hypervisor	%CPU	19.16	106.20	70.06	215.50
	RES/MB	1098.88	1105.92	1048.71	1086.60

For CPU utilization, Table 3 shows that SEPA always has a smaller average and peak CPU usage, with the exception of verifying all VMs at once. HYPA requires less CPU time for attestation verifications as it performs fewer attestation operations than SEPA.

The memory consumption depicted in Table 3 shows unexpected behavior. HYPA is in most cases more efficient than SEPA. In theory, HYPA should require more memory at verification time, as it attests both hypervisor and VMs at the same time. We can only explain this behavior with JAVA's garbage collection, or the general OS memory management messing up our measurements.

Table 2 shows that SEPA utilizes the network more efficiently until the threshold of attestation requests is reached. If more client VMs were added to the hypervisor host, or more attestations were requested at once, HYPA would become more efficient.

We also tested whether or not the integrity of VMs or underlying hypervisor hosts, influence the results of our measurements. However, we could not find correlations or indicators that non-integer systems are more expensive. The results are so close and indistinguishable, that the differences are probably caused by different sources of noise, causing jitter on the measurements, such as network communication, host utilization by other applications and services, Java Just in Time (JIT) optimizations, garbage collection, or OS scheduling.

We also monitored the Trust Agent on the clients. We could not observe any apparent differences between the schemes. CPU utilization and memory consumption showed no difference between the schemes. This may again be caused by JAVA's overhead and garbage collection. Logically, there should be slightly more memory consumption when the Attestation Manager gathers attestation information in the form of SMLs and PCR values (which need to be loaded into memory).

Nonce generation time differs for both schemes. HYPA requires more time to gather and combine all attestation information. It is overall stable and has little only impact on the overall performance of the schemes. Nonce generation

time for HYPA increases with the number of running VMs. Thus, HYPA may run into scalability issues on the hypervisor Trust Agent. But only if there are many guests and very big SMLs. SEPA only has scaling issues when many hosts need to be attested on the network, but not on a single Trust Agent.

We conclude that SEPA performs better than HYPA for a small number of VMs. Performance of SEPA decreases as the number of VM attestations increases. As expected, HYPA shows constant performance across all tests. Having support for both of the attestation schemes on target systems, efficient combinations can be defined.

8 Conclusions and Future Work

In this paper we analyzed existing Trusted Computing based attestation approaches for virtualized environments. We discovered a conceptual security weakness in the existing *separate attestation* approach.

By proposing *secure* separate attestation, we strengthened layer linking between hypervisor and VM layer. As a result, we improved security characteristics of the original separate attestation approach and provided an effective countermeasure against relay attacks. We implemented and evaluated the hypervisor-based attestation approach as well as our secure separate attestation approach on production-grade hardware. With performance measurements we underlined the feasibility of our approach for systems running a limited number of VMs. Further, we pointed out that our security improvement allows for keeping standard attestation protocols in place, by requiring only minimal changes to hypervisor software.

Future directions of work include the application of secure separate attestation in virtualized network equipment, fog, and edge computing. With the evolution of container technology the applicability of our approach beyond full-fledged machine virtualization is another interesting direction of research.

Acknowledgements

The work and results presented in this paper were developed in the scope of a technical research cooperation project on cloud integrity verification, together with the German Research Center of Huawei Technologies. We thank Huawei for giving us the opportunity to work and evaluate on production-grade hardware.

References

1. Azab, A.M., Ning, P., Sezer, E.C., Zhang, X.: Hima: A hypervisor-based integrity measurement agent. In: Computer Security Applications Conference, 2009. ACSAC '09. Annual. pp. 461–470 (Dec 2009)
2. Berger, S., Cáceres, R., Goldman, K.A., Perez, R., Sailer, R., van Doorn, L.: vtpm: Virtualizing the trusted platform module. In: Proceedings of the 15th Conference on USENIX Security Symposium - Volume 15. USENIX-SS'06, USENIX Association, Berkeley, CA, USA (2006)

3. Berger, S., Goldman, K.A., Pendarakis, D., Safford, D., Valdez, E., Zohar, M.: Scalable attestation: A step toward secure and trusted clouds. In: Cloud Engineering (IC2E), 2015 IEEE International Conference on. pp. 185–194 (March 2015)
4. Celesti, A., Fazio, M., Villari, M., Puliafito, A., Mulfari, D.: Remote and deep attestations to mitigate threats in cloud mash-up services. In: Computer and Information Technology (WCCIT), 2013 World Congress on. pp. 1–6 (June 2013)
5. Champagne, D., Lee, R.B.: Processor-based tailored attestation. Princeton University Department of Electrical Engineering Technical Report (2010)
6. Chen, W.Z., Zhang, Z.P., Yang, J.H., He, Q.M.: Cerberus: A novel hypervisor to provide trusted and isolated code execution. In: Information Science and Management Engineering (ISME), 2010 International Conference of. vol. 1, pp. 330–333 (Aug 2010)
7. Coker, G., Guttman, J., Loscocco, P., Herzog, A., Millen, J., O'Hanlon, B., Ramsdell, J., Segall, A., Sheehy, J., Sniffen, B.: Principles of remote attestation. Int. J. Inf. Secur. **10**(2), 63–81 (Jun 2011)
8. Cooperation, I.: Building Trust and Compliance in the Cloud with Intel Trusted Execution Technology - The Taiwan Stock Exchange Corporation Develops a Secure Cloud Infrastructure. Tech. rep., Intel Cooperaion (2013), `https://www.hytrust.com/uploads/2015/08/intel_txt.pdf`
9. Dolev, D., Yao, A.: On the security of public key protocols. IEEE Transactions on Information Theory **29**(2), 198–208 (Mar 1983)
10. Garfinkel, T., Pfaff, B., Chow, J., Rosenblum, M., Boneh, D.: Terra: A virtual machine-based platform for trusted computing. In: Proceedings of the Nineteenth ACM Symposium on Operating Systems Principles. pp. 193–206. SOSP '03, ACM, New York, NY, USA (2003)
11. Ghosh, A., Sapello, A., Poylisher, A., Chiang, C.J., Kubota, A., Matsunaka, T.: On the feasibility of deploying software attestation in cloud environments. In: 2014 IEEE 7th International Conference on Cloud Computing. pp. 128–135 (June 2014)
12. Lauer, H., Kuntze, N.: Hypervisor-based attestation of virtual environments. In: The 13th IEEE International Conference on Advanced and Trusted Computing (Jul 2016)
13. McCune, J.M., Li, Y., Qu, N., Zhou, Z., Datta, A., Gligor, V., Perrig, A.: Trustvisor: Efficient tcb reduction and attestation. In: 2010 IEEE Symposium on Security and Privacy. pp. 143–158 (May 2010)
14. Stumpf, F., Eckert, C.: Enhancing trusted platform modules with hardware-based virtualization techniques. In: 2008 Second International Conference on Emerging Security Information, Systems and Technologies. pp. 1–9 (Aug 2008)
15. Szefer, J., Lee, R.B.: Architectural support for hypervisor-secure virtualization. In: Proceedings of the Seventeenth International Conference on Architectural Support for Programming Languages and Operating Systems. pp. 437–450. ACM, New York, NY, USA (2012)
16. Trusted Computing Group: Virtualized Trusted Platform Architecture Specification, specification version 1.0, revision 0.26 edn. (Sep 2011)
17. Yu, A., Qin, Y., Wang, D.: Obtaining the integrity of your virtual machine in the cloud. In: Cloud Computing Technology and Science (CloudCom), 2011 IEEE Third International Conference on. pp. 213–222 (Nov 2011)
18. Zhang, T., Szefer, J., Lee, R.B.: Security verification of hardware-enabled attestation protocols. In: Microarchitecture Workshops (MICROW), 2012 45th Annual IEEE/ACM International Symposium on. pp. 47–54 (Dec 2012)