# Virtual TPM for a secure cloud: fallacy or reality?

Jordi Cucurull
Research and Security department,
Scytl Secure Electronic Voting
Email: jordi.cucurull@scytl.com

Sandra Guasch
Research and Security department,
Scytl Secure Electronic Voting
Email: sandra.guasch@scytl.com

***Abstract*—The cloud technology has dramatically increased the virtualisation usage during the last years. Nevertheless, the virtualisation has also imposed some challenges on the security of the cloud. A remarkable case is in the usage of cryptographic hardware such as the Trusted Platform Module (TPM).**

**A TPM is a device, physically attached to a server, that provides several cryptographic functionalities to offer a foundation of trust for the running software. Unfortunately, the virtualisation of the TPM to bring its security properties to virtual environments is not direct due to its design and security constraints.**

**During the last years several proposals have been presented to solve the virtualisation of the TPM. Nevertheless, the virtualisation systems have not started to adopt them until very recently. This paper reviews three existing implementations of virtual TPM in the Xen and QEMU virtualisation solutions. The main contribution of the paper is an analysis of these solutions from a security perspective.**

***Palabras clave*—cloud, security, TPM, vTPM, virtualisation, XEN, QEMU**

## I. Introduction

The cloud technology has dramatically increased the usage of virtualisation during the last years. Virtualisation has detached the software applications from the physical machines where they are hosted. This allows a sensible use of resources, since the same infrastructure can be shared by many applications and the number of running servers can be adapted to the load. Nevertheless, the virtualisation has also introduced new challenges to the security of the clouds. A remarkable case is the one where the machines were using secure hardware to ensure the integrity of the infrastructure, such a Trusted Platform Module (TPM).

A TPM [1] is a device, physically attached to a server, that provides different cryptographic functionalities to facilitate the creation of a foundation of trust of the software installed in the server. Unfortunately, the TPM was not designed with virtualisation in mind, hence its virtualisation is not direct and it implies several security considerations.

Since Berger et al. [2] presented their work about virtualisation of the TPM, a few other proposals have appeared [3], [4], [5], [6]. Nevertheless, existing virtualisation software did not seem to adopt any of these solutions until very recently.

The main purpose of this paper is the analysis of three identified implementations of virtual TPM (vTPM) for the Xen and QEMU virtualisation solutions. The paper is organised in seven sections. Section II presents the technologies related to the implementations analysed, Section III describes the vTPM solution for Xen, Section IV describes two vTPM solutions for QEMU, Section V analyses the security of the solutions presented, Section VI discusses the security findings, and Section VII presents the final conclusion of the paper.

## II. Technologies

This section describes the technologies that might be involved in a virtualised Trusted Platform Module solution.

### A. Platform virtualisation

Platform virtualisation is the practise of emulating one or more physical hosts, or parts of them, within an actual physical host. The software that creates and manages the virtual guests (emulated hosts) within the host machine (physical host) is the hypervisor. Two types of virtualisation can be distinguished:

- **Full virtualisation**: The physical host is fully emulated. The operating system of the virtual guest does not realise is running on an emulated device and it does not require any modification. This solution can be based only on software or leverage specific hardware virtualisation extensions of the CPU [7], which provide different performance.
- **Paravirtualisation**: The physical host is emulated with selected modifications of its architecture to enhance the scalability, performance and simplicity of the solution [8]. The operating system of the virtual guest has to be adapted to work with the emulated host.

In this article we selected the Xen [9], [10] and QEMU [11], [12] virtualisation systems. Xen is a system that supports full and paravirtualised x86 guests. Xen maps the virtual guests as domains. There is a privileged domain, called Dom0, and user domains, called DomU. Additionally, some of the functionality of Dom0 was disaggregated in Stub Domains [13] for security and scalability purposes. QEMU is a fully virtualised system that can emulate different architectures. As opposed to Xen that manages the whole host machine, QEMU is a standalone application within the host machine.

### B. Trusted Platform Module

A Trusted Platform Module (TPM) [1] is a device, physically attached to a server and with a standard interface called TPM-TIS [14], that provides different cryptographic functionalities in the host, e.g. to ensure the integrity of the platform. A TPM includes a Root of Trust for Storage (RTS) for external secure key storage, non volatile protected storage (NVRAM), facilities to digitally sign data and the Platform

ISBN: 978-84-9717-323-0

Configuration Registers (PCR) to store measurements of the system done by the TPM.

A TPM has at least 16 PCR registers, which are initialised to a known value when the machine is rebooted. The values of these registers cannot be arbitrarily set. Instead, they are modified by an operation called extension that performs a hash, a SHA1 in TPM 1.2 [1], of the previous value of the register and the piece of data to measure. The signed values of the PCR registers can be retrieved from the TPM by issuing the TPM Quote operation.

TPM is based on Public Key Infrastructure (PKI). The TPM has a special key, the Endorsement Key (EK), that is created by the TPM manufacturer and that can include a certificate issued by it. When the TPM is initialised by the user, in the process of taking the TPM ownership, the Storage Root Key (SRK) is generated. This key is the root of the hierarchy of keys that will be subsequently generated and used in the TPM. Finally, the Attestation Identity Keys (AIK) are used as an alias of the EK for signing information produced by the TPM, e.g. the PCR register values issued after the TPM Quote operation.

The TPM is mainly used to create a foundation of trust of the software installed in the host where the device is present. This is performed through a process called Static Root of Trust for Measurement (S-RTM) [15]. This process performs a chain of measurements, starting when the host platform is reset, of the components and configuration data involved in the system boot. Each component measures the next component before passing the control to it, forming what is called a Chain of Trust (CoT). The CoT, at least, involves the BIOS, the boot loader and the operating system kernel. The resulting measurements after a system boot, must be always the same unless the boot components are modified.

The combination of the TPM Quote operation and the S-RTM process, allows the remote attestation [16] of the host. An external attester can request a TPM Quote of the PCRs, and compare the obtained values with a baseline of the PCR values of the system generated when it was in a trusted state.

### C. Virtual TPM

Virtualisation is currently an extended practise, but the TPM was not designed for virtualised systems. The security offered by a TPM is based on the principle of a trusted piece of hardware to create the foundation of trust in a given host. The implementation of a virtual TPM (vTPM) for a virtualised environment, to provide equivalent security than a physical TPM (pTPM), requires a special care with: protection of the vTPM secrets, link between the vTPMs and the virtual guests, extension of the CoT from the host machine to the virtual guests and key management. Several works analyse and proposes solutions to the virtualisation of the TPM [2], [17] and to its integration in virtualised systems [3], [4], [5], [6]. Nevertheless, it is not until recently that implementations have started to come up and be integrated in well-known virtualised environments (see Sections III and IV).

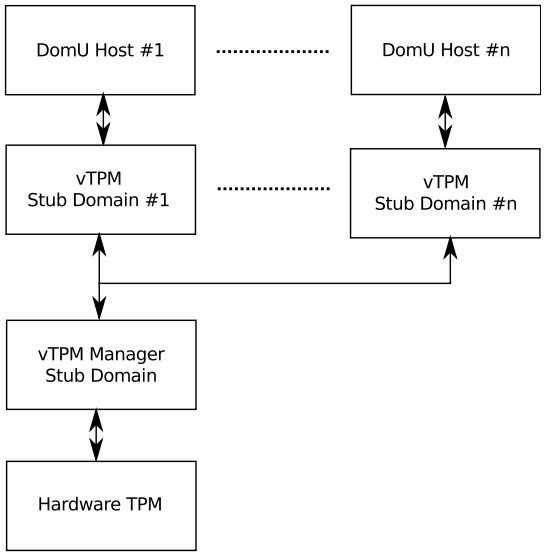


Figure 1. Structure of vTPM service in Xen

## III. Virtual TPM in Xen

Xen 4.3 implements the service of virtual TPM (vTPM) only for paravirtualised guests.The service is designed as a set of secure separate stub domains (see Figure 1) managed by the system hypervisor, each of them running a mini-os [18] and its dedicated functionality. Each virtual guest has a software emulated TPM, based on the TPM Emulator [19], running in a vTPM stub domain. And there is a vTPM Manager stub domain that coordinates and links the vTPMs with the physical TPM (pTPM). The most relevant characteristics of the virtualised TPM implementation of Xen are:

- **Non transparent vTPM**: A custom kernel module driver (tpmfront) must be installed in each virtual guest. The module provides the standard TPM interface (`/dev/tpm`) to the applications of the virtual guest, i.e. they can use the vTPM as if it was a pTPM. The custom kernel module driver is not integrated in the current Linux kernels and it is not easy to find. In addition, the driver is not available for non Linux based operating systems.
- **vTPM's secrets bound to pTPM**: The secrets of the vTPM are encrypted with AES-256 and stored in disk. The symmetric key is bound to an storage RSA key of 2048 bits. The RSA key is generated by the pTPM and can only be used by it.
- **Configurable TPM ownership and SRK authentication**: The passwords used to access the pTPM and the SRK are configurable. These passwords must be provided at the time of loading the vTPM Manager stub domain.
- **Passthrough of certain Physical TPM registers**: The administrator of the vTPM stub domain can configure certain PCR registers of the vTPM to adopt the values of the same registers of the pTPM.
- **Extension of CoT from the host machine to the virtual guests**: If the "pv-grub" external bootloader is used to boot the virtual guest, the guest kernel is measured

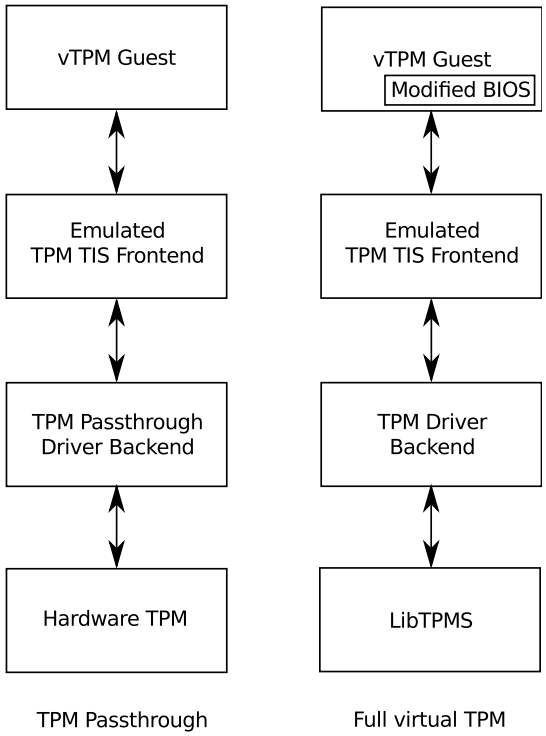


Figure 2. Structure of vTPM services in QEMU

in PCR #4 and the boot command line and initrd are measured in PCR #5 of the vTPM. Hence, the integrity of the guests can be ensured if the "pv-grub" bootloader, the hypervisor and other components that support the guests are trusted.

- **Migration of vTPM not supported**: All virtual TPMs are bound to a specific pTPM. Hence, guests with virtual TPMs cannot be migrated to another physical server.

## IV. Virtual TPM in QEMU

QEMU supports virtual TPMs (vTPM) for guests from its version 1.5. Nevertheless, the only officially supported vTPM is based on TPM passthrough. This means that the TPM offered to the virtual guests is the actual physical TPM (pTPM) of the physical host. In addition, there is an implementation of the vTPM that is not officially integrated into QEMU that provides full vTPMs.

### A. TPM passthrough

The TPM passthrough, as previously mentioned, provides a vTPM to the virtual guests which is a direct mapping with the pTPM of the physical machine. The service is designed as a backend driver for the pTPM that communicates with an emulated TPM TIS frontend (see Figure 2). The most relevant characteristics of this implementation are:

- **Transparent vTPM offered to guest host**: The virtual guest sees the vTPM as if it was a pTPM. No special kernel drivers are needed.
- **Passthrough of all the physical TPM registers**: The vTPM is a direct mapping of the pTPM. The PCR values, NVRAM area and keys of the vTPM are the same of the pTPM. Hence, all the measurements performed by the physical host are reflected to the vTPM.
- **Only one virtual guest can be provided with vTPM**: The reason is the one to one mapping between the vTPM and the pTPM. The registers and the NVRAM of the pTPM cannot be multiplexed to support multiple vTPMs.
- **Migration of vTPM is not supported**: The pTPM registers and NVRAM cannot be extracted from the TPM and imported into another pTPM. Hence, migration cannot be supported.

### B. Full virtual TPMs

The full virtual TPMs approach, as previously mentioned, provides a complete vTPM implementation to the virtual guests that is totally detached from a pTPM. The service is designed as a software TPM backend implementation linked with the external library libTPMS. This library provides TPM emulation. On the guest side there is an emulated TPM TIS frontend (see Figure 2) and a modified open source BIOS, based on SeaBIOS [20], to support the vTPM. The most relevant characteristics of this implementation are:

- **Transparent vTPM offered to guest host**: The service of vTPM is based on full TPM emulation. Since the TPM TIS interface is emulated, no modifications have to be performed to the guest operating system.
- **vTPM's secrets stored into QEMU image**: The secrets of the vTPM are stored within an image file. The secrets are not encrypted by default, however QEMU allows the use of encrypted images, e.g. QCOW2 [21] can provide AES-128 encryption.
- **No pTPM required**: Since the vTPMs are fully emulated and not bound to a pTPM, this solution does not require the presence of a pTPM in the system.
- **Modified BIOS with vTPM and SRTM support**: A set of patches [1] to be applied to SeaBIOS are provided. The patches add vTPM support and implement the Static Root of Trust for Measurement (SRTM), i.e. the code that takes care of the first measurements right after the machine is powered on.
- **Migration ready**: The migration of the vTPM is not implemented, but it would not be difficult to integrate because the vTPM is not strongly linked to a pTPM.

## V. Security considerations

There are four aspects of the vTPMs in virtualised environments that define their level of security regarding a pTPM: protection of the vTPM secrets, link between the vTPMs and the virtual guests, extension of the CoT from the host machine to the virtual guests and key hierarchies and management. This section analyses these four aspects for the vTPM implementations presented (see also Table I).

[1] See e-mail with patches of Stefan Berger "[PATCH V3 0/8] Add TPM support to SeaBIOS" of April 2011 in SeaBIOS mail list

Table I
COMPARISON OF VTPMS

| | Xen | QEMU TPM Passthrough | QEMU Full vTPM |
|---|---|---|---|
| Approach | Multiple vTPMs | pTPM passthrough | Multiple vTPMs |
| Multiple virtual guests | Yes | No | Yes |
| Transparent vTPM | No | Yes | Yes |
| vTPM secrets | Digital envelope linked to pTPM (AES-CBC 256 bits and RSA 2048 bits) | pTPM | Image (allows AES-CBC 128 bits) |
| Link of vTPM and virtual guest | Weak | Static | Strong (state) / Weak (secrets) |
| Physical to virtual Chain of Trust | No, but includes external bootloader that measures kernel | No | No, but includes external BIOS with SRTM |
| Key hierarchies | Independent | Keys of pTPM | Independent |
| VM Migration | No | No | Yes |
| Orientation | Production | Development and testing | Production |
| Implementated in | Xen Project 4.3 and above | QEMU 1.5 and above | Experimental patches for development |

### A. Protection of the vTPM secrets

A TPM has data that must be kept secret and safe from manipulation, e.g. the Endorsement Key (EK) or the data contained in the NVRAM area.

The vTPM secrets in Xen are bound to the pTPM through a digital envelope. The data of the envelope is ciphered with AES-CBC symmetric encryption with a 256 bits key generated using the pTPM TRNG [22]. The symmetric key is protected with a public key of the pTPM. Hence the vTPM secrets can only be recovered accessing the pTPM. It is announced that in the future there will be the possibility to seal the symmetric key, i.e. the current protection linked to the PCR values of the pTPM. In that case if the hypervisor or Dom0 critical elements are corrupted, due to a change of the PCR values, the vTPM secrets will not be available.

In QEMU TPM Passthrough the vTPM secrets and registers are literally protected by the pTPM. This has the advantage of the hardware-based security offered by the pTPM, but it also means that anybody with access to the pTPM has access to the vTPM secrets.

In QEMU Full vTPM the secrets are kept in a dedicated image file without any protection mechanism implemented. Nevertheless, it is possible to leverage the security offered by the specific type of image used. Currently only QCOW2 offers privacy, in this case password based encryption with AES-CBC and 128 bits key. Nevertheless, the password is limited to 16 alphanumeric characters, hence its security level is limited to 105 bits. No authenticated encryption [23] nor other integrity-preserving mechanisms are used, hence the secrets could be manipulated. In addition, the system is not mature enough and it was failing when a QCOW2 encrypted image was used.

### B. Link between vTPMs and virtual guests

The link between vTPMs and virtual guests must be protected. Otherwise, a virtual guest could be provided with a different, and probably manipulated, vTPM with measurements that may not correspond to the guest.

In Xen, the vTPM is completely independent of the virtual guest, including their lifecycles, and they run in different domains. The link between the vTPM stub domains and the virtual guest domains is not robust neither authenticated. Hence any vTPM domain can be linked to any virtual guest domain. In addition, it is possible to pause a virtual guest domain and replace its vTPM domain with the one of another guest, i.e. completely replacing PCR registers and non volatile data. This allows a corrupt admnistrator, or attacker with equivalent privileges, in Dom0, to manipulate the vTPM virtual guests association.

In QEMU TPM Passthrough the association between vTPM and virtual guest is static, since there is only one possible vTPM that is mapped to the pTPM. Despite this increases the security of the vTPM secrets in front of attackers without privileged rights, the vTPM lifecycle and state are mapped to the pTPM. Hence, anyone with access to the pTPM or to the host node can manipulate the measurements shown in the vTPM, e.g. by directly accessing the pTPM, enabling another virtual guest with access to it or rebooting the virtual guest (on reboot of the virtual guest the vTPM values are not initialised since the lifecycle of the vTPM are linked to the physical machine).

In QEMU Full vTPM, the vTPM is implemented and managed by the same instance of the hypervisor that manages the virtual guest. Hence the association with the vTPM and virtual guest lifecycle is implicit, i.e. there is no possibility to manipulate the PCR registers. Nevertheless, there is no strong link between the image file that contains the vTPM secrets and its virtual guest.

### C. Chain of Trust extension to the virtual guests

In a virtualised TPM solution, the security offered by the vTPMs depends on the underlying host machine. It is desirable to create a Chain of Trust (CoT) in this host and link it to the individual CoTs created in each virtual guest. The verification of the CoT extension requires access to the measurements of both pTPM and vTPM to evaluate.

In Xen, the "pv-grub" guest bootloader allows the extension of the CoT from the host machine to the virtual guests. In detail, the bootloader measures, in the vTPM, the kernel, initrd and command line used to boot the guest. Additionally, the bootloader can be measured in one of the registers of the pTPM and, this register, be selected to be shown as one of the vTPM PCRs. In this case, both CoT of the host machine and virtual guest would be linked. Nevertheless, the authors of the solution discourage the direct usage of the pTPM in the host machine. The reason is that if the pTPM drivers and software stack are installed in Dom0, the administrators of the system have easy access to the pTPM and, as a consequence, to the key used to encrypt the data of the vTPMs. This prevents the usage of the pTPM for measuring and checking the integrity of Dom0 and extending the CoT. Nevertheless, this will be solved in the new-coming releases of the solution.

In QEMU TPM Passthrough, it is not possible to extend the CoT of the physical host to the virtual guest since the PCR registers of the pTPM and vTPM are the same, the lifecycle of the vTPM is linked to the one of the pTPM and the bootloader of the guest image cannot be measured by QEMU. In this solution there is no clear border between the security of the physical host and the security of the virtual guest.

In QEMU Full vTPM, the modified BIOS provides support to link the CoT of both the node machine and the virtual guests. The modified SeaBIOS implements the S-RTM process, which allows to create a CoT within the the virtual guest. If the BIOS of the virtual guests, the hypervisor and other software managing the system is measured in the host machine, the link between the host machine and virtual guests CoT can be created.

In all the cases where the CoT extension would be possible, the system is vulnerable to malicious administrators replacing the bootloaders during runtime, virtual guest BIOS, or any other software involved in the virtual guest management.

### *D. Key hierarchies and management*

All TPMs have at least an EK and, after its ownership is taken, a SRK which is the root for its key hierarchy.

In the full vTPM implementations in Xen and QEMU the keys are completely independent of the ones present in the pTPM. Despite this implies a loosely coupled key hierarchy with a pTPM, in practise will facilitate the migration of the vTPMs when this becomes ready in the future. In Xen the EK is automatically generated the first time the vTPM is initiated, while in QEMU the EK has to be explicitly generated by the user issuing a specific command from within the virtual guest. In the QEMU TPM Passthrough implementation, the keys used in the vTPM are the same used in the pTPM.

Additional options exist, when the key hierarchy of a vTPM is generated [2], in order to provide keys that may become certified by a certificate authority. Nevertheless, the current vTPM implementations still do not offer them.

Regarding the key generation, in Xen and QEMU TPM Passthrough the pTPM TRNG is used as random number generator. While the QEMU Full vTPM implementation uses a random number generator provided by the OpenSSL library.

## VI. Discussion

Given the security considerations detailed in Section V, it can be stated that the security currently provided by the existing vTPMs implementations is not equivalent to the security of a pTPM. In all the cases, the security of the virtual guests depend on the administrators of the machine hosts. Nevertheless, the fully virtualised vTPMs of Xen and QEMU set the bases for a near future usage of this technology.

In Xen, if there is a malicious administrator in the physical host, the security offered by the vTPM of the virtual guests cannot be guaranteed. This is something known by the authors of this implementation [2]. As they state, the solution is to create a domain building component measured by the pTPM during boot. This component should have a static library with the critical domains to build. This component should enforce the creation and destruction of these domains as well as the correct pairing of vTPM domains and guests. We believe that an administrator should not be allowed to log into the machine without modifying the measurements of the TPM, e.g. the login could add a measurement to the TPM of each user that logs into the system. This could be used as a tamper-proof mechanism. Hence the physical machine would become a kind of sealed box.

In QEMU, the difference with Xen is that there is no hypervisor that controls the whole virtualised system. In this case, for a maximum security, a software manager of the virtual guests should be installed in the physical host. The manager should ensure the image file that contains the vTPM secrets and the modified BIOS were correctly paired to the correct virtual guest to ensure its integrity. This manager could be measured as part of the physical machine CoT. In this case it would also be possible to extend the CoT of the physical host to the virtual guest, assuming the patched SeaBIOS is in place and a secure bootloader is installed in the virtual guest. Since the BIOS code used in QEMU is explicitly provided when the guest is started, the tool that manages the guest machines could ensure its integrity. As in Xen, the physical host could generate measurements in the TPM for each user logged, hence it would become as a sealed box in the sense that nobody can log to perform system changes without being detected.

## VII. Conclusions

In this article we have analysed two virtualisation solutions with three currently available virtual TPM approaches. The purpose of this analysis was to determine if there where available virtualised TPM solutions and the level of security offered by them.

After the presented analysis we found two implementations that offer complete TPM virtualisation for Xen and QEMU. The implementation in Xen still has not reached a level of

[2] See "Questions about the usage of the vTPM implemented in Xen 4.3" in February 2014 in the xen-devel mailing list.

security comparable to a non virtualised solution, but their developers are pushing hard for it. In addition, the solution is integrated within the Xen official releases. The implementation in QEMU offers less security than the one in Xen, e.g. to store the secrets of the vTPM, and its integration with QEMU is not officially supported due to restrictions of the project for including code that has dependencies with external libraries (in this case because of the libTPMS).

Given the development activity seen, it is expected the improvement of the security and availability of the virtualised TPM solutions soon. In addition, the virtualised systems will integrate other technologies that enhance the trust with their hypervisor, e.g. the support of the IntelTXT technology [24] that simplifies the foundation of trust for the hypervisors in virtualised systems in conjunction with the TPM.

## Acknowledgements

We want to thank the developers of the Xen and QEMU vTPM solutions for the information given through the different development forums. This work has been co-funded by the project Trusted Cloud IPT-2011-1166-430000 of the Ministry of Economy and Competitiveness (MINECO) and the European Fund for Regional Development (FEDER)".

## References

[1] TCG, "TPM Main Specification Level 2 Version 1.2," March 2011.
[2] S. Berger, R. Cáceres, K. A. Goldman, R. Perez, R. Sailer, and L. van Doorn, "vTPM: Virtualizing the Trusted Platform Module," in *Proceedings of the 15th Conference on USENIX Security Symposium - Volume 15*, ser. USENIX-SS'06. Berkeley, CA, USA: USENIX Association, 2006.
[3] S. Berger, R. Cáceres, D. Pendarakis, R. Sailer, E. Valdez, R. Perez, W. Schildhauer, and D. Srinivasan, "TVDc: Managing security in the trusted virtual datacenter," *SIGOPS Oper. Syst. Rev.*, vol. 42, no. 1, pp. 40–47, Jan. 2008.
[4] B. Danev, R. J. Masti, G. O. Karame, and S. Capkun, "Enabling secure VM-vTPM migration in private clouds," in *Proceedings of the 27th Annual Computer Security Applications Conference*, ser. ACSAC '11. New York, NY, USA: ACM, 2011, pp. 187–196.
[5] D. Liu, J. Lee, J. Jang, S. Nepal, and J. Zic, "A cloud architecture of virtual trusted platform modules," in *Embedded and Ubiquitous Computing (EUC), 2010 IEEE/IFIP 8th International Conference on*, Dec 2010, pp. 804–811.
[6] D. Wallom, M. Turilli, G. Taylor, N. Hargreaves, A. Martin, A. Raun, and A. McMoran, "myTrustedCloud: Trusted cloud infrastructure for security-critical computation and data managment," in *Cloud Computing Technology and Science (CloudCom), 2011 IEEE Third International Conference on*. IEEE, 2011, pp. 247–254.
[7] K. Adams and O. Agesen, "A comparison of software and hardware techniques for x86 virtualization," *SIGARCH Comput. Archit. News*, vol. 34, no. 5, pp. 2–13, Oct. 2006.
[8] A. Whitaker, M. Shaw, and S. D. Gribble, "Denali: Lightweight virtual machines for distributed and networked applications," in *In Proceedings of the USENIX Annual Technical Conference*, 2002.
[9] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield, "Xen and the art of virtualization," *ACM SIGOPS Operating Systems Review*, vol. 37, no. 5, pp. 164–177, 2003.
[10] "Xen Project," http://www.xenproject.org.
[11] F. Bellard, "QEMU, a fast and portable dynamic translator," in *USENIX Annual Technical Conference, FREENIX Track*, 2005, pp. 41–46.
[12] "QEMU," http://wiki.qemu.org.
[13] S. Thibault, "Stub domains: A step towards dom0 disaggregation," Xen Summit, 2008, http://blog.xen.org/index.php/2008/08/28/xen-33-feature-stub-domains/.
[14] TCG, "PC Client Work Group PC Client Specific TPM Interface Specification (TIS), Version 1.3," March 2013.
[15] TCG, "PC Client Work Group Specific Implementation Specification for Conventional Bios Specification, Version 1.2," February 2012.
[16] G. Coker, J. Guttman, P. Loscocco, A. Herzog, J. Millen, B. O'Hanlon, J. Ramsdell, A. Segall, J. Sheehy, and B. Sniffen, "Principles of remote attestation," *International Journal of Information Security*, vol. 10, no. 2, pp. 63–81, 2011.
[17] F. Stumpf and C. Eckert, "Enhancing trusted platform modules with hardware-based virtualization techniques," in *Emerging Security Information, Systems and Technologies, 2008. SECURWARE'08. Second International Conference on*. IEEE, 2008, pp. 1–9.
[18] S. Popuri, "A tour of the mini-os kernel," http://www.cs.uic.edu/~spopuri/minios.html.
[19] M. Strasser and H. Stamer, "A software-based trusted platform module emulator," in *Trusted Computing - Challenges and Applications*, ser. Lecture Notes in Computer Science, P. Lipp, A.-R. Sadeghi, and K.-M. Koch, Eds. Springer Berlin Heidelberg, 2008, vol. 4968, pp. 33–47.
[20] "SeaBIOS," http://www.seabios.org/SeaBIOS.
[21] M. McLoughlin, "The QCOW2 image format," September 2008, https://people.gnome.org/~markmc/qcow-image-format.html.
[22] A. Suciu and T. Carean, "Benchmarking the true random number generator of TPM chips," *CoRR*, vol. abs/1008.2223, 2010.
[23] M. Bellare and C. Namprempre, "Authenticated encryption: Relations among notions and analysis of the generic composition paradigm," in *Advances in Cryptology - ASIACRYPT 2000*, ser. Lecture Notes in Computer Science, T. Okamoto, Ed. Springer Berlin Heidelberg, 2000, vol. 1976, pp. 531–545.
[24] Intel Corporation, "Intel TXT software development guide," 2014, http://download.intel.com/technology/security/downloads/315168.pdf.