A Trusted Way for Encryption Key Management in Cloud Computing

Saad Fehis, Omar Nouali, and M-Tahar

***Abstract*—We propose an approach to provide the cryptography key management system (CKMS) as a trusted security services in Cloud Computing, based on the trusted platform module (TPM / vTPM). In this approach we have used the TPM's capabilities / functions as a secure way and a root of trust for this kind of services. Therefore, and as an application case, we have used TPM's key generation component as a trusted way to generate and to sign an encryption/signing keys by the CKMS for their customers.**

***Index Terms*—Cloud Computing, Security as a Services, Cryptographic Key Management System, Trusted Platform.**

I. Introduction

Providing an IT services today, favoured by the evolution in computing architecture known by a Cloud Computing (*IaaS, PaaS, SaaS*)[1]. Consequently, the number of cloud-based services' users have been increasing, (*such as salesforce.com or Google Apps,*) means that many mobile IT users will be accessing business data and services without traversing the corporate network. This will increase the need for enterprises to place security controls between mobile users and cloud-based services. Therefore, the apparition of new offers of a Security as a services (*SecaaS*). The SecaaS refers to the provision of security applications and services via the cloud either to cloud-based infrastructure and software or from the cloud to the customers' on-premise systems. This will enable enterprises to make use of security services in new ways, or in ways that would not be cost effective if provisioned locally [2].

Gartner is predicting the cloud-based security services market, which includes secure email or web gateways, identity and access management (IAM), remote vulnerability assessment, security information and event management to hit 4.13 billion by 2017 [3]. According to its "Market Trends: Cloud-based Security Services Market, Worldwide, 2014," Gartner is predicting growth is likely to come because of the adoption of these cloud-based security services by small-to-mid-sized business (*SMB*) in particular. Certain market segments mentioned in the report will see higher overall sales and year-over-year growth.

However, the shared resources (*multi-tenant environment*) create many security issues, therefore, a real challenge in the adoption of Cloud-Computing [4]. Therefore, there are a new problems add to the traditional threats, will be addressed such as: vulnerability due to virtualization of the physical infrastructures [5], data isolation, management of privacy and confidentiality of data. Consequently, the creation of the trust between the customers and the services provider.

In this work, we have focused on the providing to the clients a trusted software components (*as a virtual Platforms*), using they for providing a security as a service (*SecaaS*)[2]. These components provide to customers a various of a security services features as: the e-mail / web content filtering, identity and access management, cryptography or cryptographic key management system (*CKMS*). The latest service is the main subject matter in this work.

The CKMS as a services, can be provided it as a dedicated or shared platforms (*to companies or users group*), and this depending on the software architecture adopted. Their security in a virtualized environment requires the assurance of its integrity at the disk level (*Source, data, executable*) and at memory level (*running*) firstly, and secondly, providing to the owner of CKMS service or a trusted third party the control, the check or the audit, by verification mechanisms of the trusted chain.

However, the safety of this type of solution, does not only depend on the integrity of software, but also of its operation and its interaction with a multi-tenant environment as; the access control (*Meta data CKMS*) and the control of information flow between instances of the same CKMS provided by the same service provider for different customers (*companies or group of users*) or with other kind of software hosted in the same cloud (*the same shared physical infrastructure using the virtualization*). In this context and to deal with those challenges, the CKMS needs to manage the encryption keys in secure way, and in all of its life cycle, from the key generation with its related data (*meta data*), to delivering they to the their customers (*owner of the key*). therefore, we need to a trusted component to answer this capability as the Trust platform Module (*TPM*)[6], (*vTPM*) [7].

In this work, we propose an approach to provide the CKMS as a trusted security services (*CKMS-SecServ*) in the cloud computing based on the virtual trusted platform module (*vTPM*). The vTPM's capabilities / functions can be used as a secure way for the protection of the CKMS-SecServ service; therefore, the vTPM used as a root of trust, and as an application case, we have used the vTPM's key generation component as a secure and a trusted way to generate and to sign any encryption/signing keys for the client for the CKMS-SecServ.

In order to treat these challenges, we have structured this paper as follows: In section 2 we will present the CKMS, its architecture, theirs security challenges, and its deployment in the Cloud Computing. In section 3, a background on the TPM, root of trust and related works. In section 4, we will present the

TPM vs CKMS_SecServ, with returned results. In the section 5 we will present a conclusion with the future direction of our work. Then the list of references used in this work.

II. Cryptographic Key Management Systems

In this section we will present firstly a set of definitions related to the CKMS. Then secondly the CKMS as service and its related to Software as service model; and finally, the security challenges related to a CKMS as a security services (*CKMS-As-a-SecServ*).

A. CKMS Presentation

A CKMS consists of policies, procedures, components and devices that are used to protect manage and distribute cryptographic keys and certain specific information, called (*associated*) metadata herein. A CKMS includes any device or sub-system that can access an un-encrypted key or its metadata. A CKMS will be a part of a larger information system that executes information processing applications. While the CKMS supports these applications by providing cryptographic key management services [8].

A CKMS can be as simple as a software program running on a single-user computer and supporting user applications. It can also be as complex as a variety of sub-systems, each containing many devices that provide key management services to numerous networked users and applications. Therefore, it may be widely distributed geographically and connected with a myriad of communications networks. [8]

A key is associated with metadata that specifies characteristics, constraints, acceptable uses, and parameters applicable to the key. For example, the key type, how it was generated, when it was generated, its owner's identifier, the algorithm for which it is intended, and its cryptoperiod. Therefore, the metadata elements may be generated by the same entity that generates the key or they may be received from another trusted entity. [8]

B. CKMS in the Cloud Computing

In Cloud computing, the CKMS as a service can be viewed as a SaaS [1]. The SaaS is "a Software deployed as a hosted service and accessed over the Internet". From an application architect's point of view, a designed SaaS application is: Scalable, Multi-tenant-efficient, and Configurable [9]. An application SaaS maturity can be expressed using a model with four distinct levels (Fig.1). Each level is distinguished from the previous one by the addition of one of the three attributes listed above.

Architecturally, SaaS applications are largely similar to other applications built using service-oriented design principles [9]. However, the real challenge is: How to keep a separate customer's data for each level, and at each data state: as in storage (*memory/disk*); running (*processor*) and in transit network? Therefore, the development and the use of a configurable metadata is not an easy task!

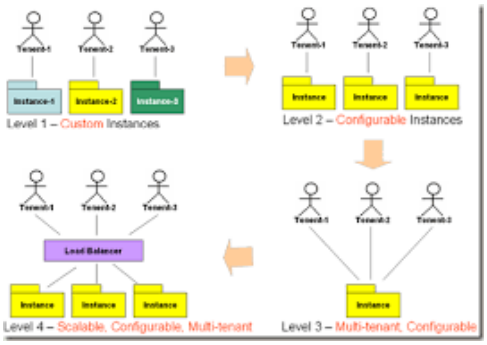

Fig. 1. Four-level Saas maturity model [9]

In our case, the CKMS as a Security Service architecture, can be developed, deployed and provided as any SaaS applications (Fig.1). However the nature of this service as security service needs a real isolation of each CKMS's instance from any other software hosted in the same shared platform physique. Therefore, the using of the second level (*configurable*) is a interesting for the isolation but not for a good performance. Consequently, using of the 4th level load balancing between the same customer's instances is the best choice (see Fig.2). And we can apply the isolation between the set of the same tenant's instances and the other tenants. This isolation can be viewed as the built of a wall around each tenant's instances.

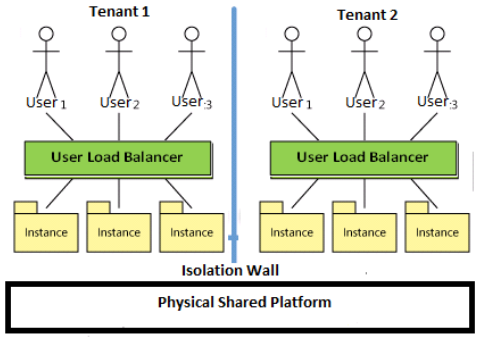

Fig. 2. Multi Tenant Isolation

C. CKMS-SecServ's Security Challenges

A CKMS must be designed in a manner that supports the goals of each organization using the CKMS. Several types of policies and their relationships will influence the design of a CKMS, as information management policy (*IMP*), information security policy (*ISP*) and CKMS security policy. A CKMS security policy is created to establish and specify requirements for protecting the confidentiality, integrity, availability, and source authentication of all cryptographic keys and metadata used by the organization. These protection requirements cover the entire key life cycle, including when they are operational, stored, and transported. A CKMS Security Policy includes the selection of all cryptographic mechanisms and protocols to be used throughout the organization's automated information systems [8].

Like keys, metadata needs to be protected from unauthorized modification and disclosure and have its source adequately authenticated. Therefore, a key and its metadata need a trusted association and supporting processes between them[8]. Therefore, a CKMS requires different types of security controls to protect its components, devices, and the data that they contain. To do this, the CKMS will likely require computer systems to perform functions in a secure way, such as key generation, key storage, key recovery, key distribution, cryptographic module control, and metadata management.

However, how protect and provide the CKMS hosted in the Cloud Computing? How perform these functions in a secure way? In this context, our work has focused on the security approaches used to provide a CKMS as a trusted components.

To protect the metadata (*CKMS's dictionary D-CKMS*) of the CKMS, we proposed an encrypted data's model for the D-CKMS [10]. The model is an encryption of the D-CKMS, that provides the data management without decryption. However, and to introducing the trust concept in this kind of service we need a root of trust based on the trusted platform module (*TPM*), where we will be presented in the next section.

III. TPM's Background

The Trusted Platform Module (*TPM*) is a hardware chip designed to enable commodity computers to achieve greater levels of security than was previously possible.The TPM stores cryptographic keys and other sensitive data in its shielded memory, and provides ways for platform software to use those keys to achieve security goals. Application software such as Microsoft's BitLocker and HP's ProtectTools use the TPM in order to guarantee security properties.

TPMs are manufactured by chip producers, including Atmel, Broadcom, Infineon, Sinosun, STMicroelectronics, and Winbond. It is specified by the Trusted Computing Group (*TCG*) and other industry consortium in three documents [6] totalling about 800 pages. In this section we will give an overview on the TPM and the trust, its architecture and its operation.

A. The TPM and the Trust

The TPM has been designed specifically to support trusted computing platforms. Therefore, in order to understand the TPM design requirements, it is first necessary to understand what the desirable features of a trusted platform are. To do this, a definition is required as to exactly what is meant by the term "trusted platform":

- The TCG defines trust to be: the expectation that a device will behave in a particular manner for a specific purpose [11],
- Another definition of a trusted platform is provided by Pearson [12] who states that: A Trusted Platform is a computing platform that has a trusted component, probably in the form of built-in hardware, which it uses to create a foundation of trust for software processes,
- or by Balacheff et al. [13] who say: A trusted platform is defined as a computing platform that has a trusted component, which is used to create a foundation of trust for software processes.

It is perhaps appropriate at this point to make a subtle distinction between what is meant by a trusted component, such as the TPM, and a trustworthy component. Anderson [14], [15] given definition of these terms who states that: "*The proper definition is that a trusted system or component is one whose failure can break the security policy, while a trustworthy system or component is one that won't fail*"

By implementing this trusted component, the TPM, as a tamper proof integrated circuit; and binding it to the platform, usually on a printed circuit board containing a more powerful processor capable of running software applications; the TPM can be used as the foundation of trust for higher level processes that run on the main processor.

In order to establish this foundation of trust, the TPM is expected to provide a fundamental set of security features which have been defined by the TCG. The features that a trusted platform should have are: Protected Capabilities, Integrity Measurement and Reporting, Confidentiality and Integrity Protection, Secure Storage and Process Isolation.

B. TPM's Components

The TPM has a set of components (Figure 3); which we classified they as flowing:

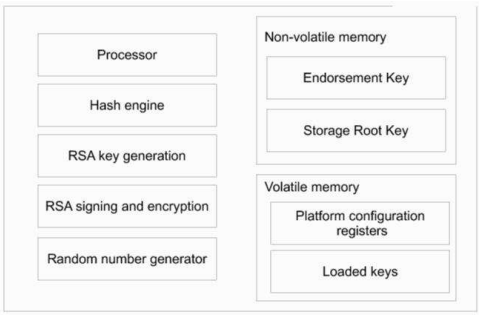

Fig. 3. TPM Architecture

- The first class related to a set of tools, for managing the encryption/signing keys and related algorithms. In this set we have the *Cryptographic Co-Processor* implements cryptographic operations within the TPM. Those operations include the Asymmetric key generation (*RSA*), Asymmetric encryption/decryption (*RSA*), Hashing (*SHA-1*) and the Random number generation (*RNG*). The TPM uses these capabilities to perform generation of random data, generation of asymmetric keys, signing and confidentiality of stored data. The TPM may symmetric encryption for internal TPM use but does not expose any symmetric algorithm functions to general users of the TPM.
- The second class related to the storage, where we have two types of memory. The first is a Non-Volatile Memory that used to store persistent identity and state associated with the TPM as the endorsement key (*EK*) and storage

root key (*SRK*); the second is a volatile memory contains the platform configuration register (*PCR*), is a 160-bit storage location for discrete integrity measurements and the loaded keys created by the TPM.

- The third class related to the Execution of the code, the powering of the TPM and the In/Out bus; where the *Execution Engine* runs program code to execute the TPM commands received from the I/O port. The execution engine is a vital component in ensuring that operations are properly segregated and shield locations are protected. The *Opt-In* component, provides mechanisms and protections to allow the TPM to be turned on/off, enabled/disabled, activated/deactivated. The Opt-In component maintains the state of persistent and volatile flags and enforces the semantics associated with these flags. the last one is the *I/O* component, it manages information flow over the communications bus.

C. TPM's keys Operations

Each TPM has a unique public/private key pair called the *endorsement key (EK)*, set at manufacture time and usually certified by the manufacturer. The EK can be taken to be the identity of the TPM. In addition to EK, when ownership of the TPM is taken, the TPM generates a public/private key pair called the *storage root key (SRK)* which is the root of the tree of storage keys; and it also generates a secret random value called *tpmProof* which is used by the TPM to identify blobs that it creates.

For platform authentication, one may create signing keys known as *application identity keys (AIKs)*. These may be used to sign (*appropriately tagged*) application-specific data, and to sign PCR values. For such signatures to be useful, AIKs need to be certified as belonging to a TPM. For reasons of user privacy, the certificate will not specify which TPM they belong to; it will just specify that they belong to a TPM. There are two ways to obtain a certificate on an AIK: using privacy CAs, or using Direct Anonymous Attestation (DAA) [6].

Now how about the measurement and reporting operations? The TPM contains a number of 160-bit registers called platform configuration registers (PCRs) intended to enable a relying party to obtain unforgeable information about the platform state. We think of the platform as consisting of several "components", which may receive control and pass on control to another component. Typical components are the BIOS, the master boot record, boot sectors, the boot loader, and ultimately the operating system and application software. A component can "measure" another component (compute its hash) and insert that measurement into a PCR (for example, before passing control to it). This insertion is an irreversible process, known as "extending" the PCR.

A given PCR can be extended with any number of measurements. The current value of the PCR represents the accumulation of them all. A *secure chain of trust* can be established by ensuring that the very first code segment executed on power-up is measured and that measurement is extended into a PCR. Then, every component A that loads another component B and passes control to it ensures that B is first measured and the measurement is extended into a PCR. The $PCRs$ then represent an accumulated measurement of the history of the executed code from power-up to the present.

A TPM signing key (*AIK*) can be used to sign the values of the $PCRs$. In this way, application software can send assurance about the state of the platform to a third party. Additionally, PCR values can be used to ensure that certain data is accessible only to authorized software.

D. Trusted Platform in the Cloud Computing

In Cloud Computing, all hardware components are shared resources by a virtualization technical. However, the TPM is designed for a single physical machine with a single operating system. Therefore, several TPM virtualization approaches and use to multiple instances of virtual TPM (*vTPM*) have been proposed, with the aim that each vTPM is designed to secure a single virtual machine [7], [16], [17], [18], [19], [20]. However the establishment of a vTPM in the virtual machine manager layer (*VMM*) may be venerable to attack different types of software, including the attacks of VMM itself (Fig.4). Therefore, securing a vTPM is directly depend on securing VMM and a virtualization techniques, which involve many challenges remain to be addressed: as the protection of vTPM at Storag, their secrets across Reboots, attestation (*Credentials, Deep Attestation Deep Attestation Layer Bindings*), vTPM backup, restore and migration.

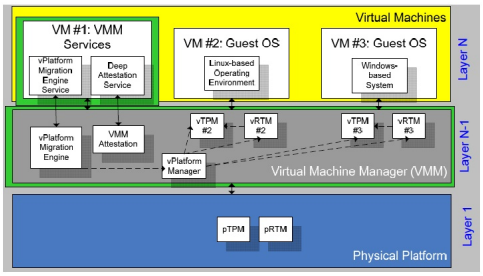

Fig. 4. Three layer architecture with privileged [7]

IV. TPM vs CKMS as a Security Services

The CKMS used to protect manage and distribute cryptographic keys and certain specific information, called (*associated*) metadata. Therefore, the CKMS needs a key generated and signed by a trusted component, and delivered it in a secure way to this CKMS, where then, the key will be delivered to he / she (*any user, thing or application*) requested the key. The key can be symmetric or asymmetric, and can be used for any purpose encryption/signing in any way as in the cloud computing, internet of things, user mobile outside of his company's network or inside of any company's network.

The Trusted platform module can be used by the CKMS as a trusted component (see Fig.5) for this purpose. However, how can use it to provide a CKMS as a trusted service in the cloud computing. In the following section we will present the TPM's capabilities for the management of the keys, then the TPM in the cloud computing.

A. TPM's Key Types and Related Functions

The TPM has a set of tools and components (Fig.3) used for the keys generation[6]. The TPM key types are defined at key creation time by the User, the different key types are:

- Non-Migratable Key (NMK): A key which is bound to a single TPM. This is a key that is (statistically) unique to a single TPM and can not be migrated or exported from the TPM.
- Migratable Key (MK): A key which is not bound to a specific TPM, and with suitable authorization, can be used outside a TPM or moved to another TPM.
- Certifiable Migratable Key (CMK): A key whose migration from a TPM is highly controlled and the TPM can attest / certify it properties

The migration destinations are defined and authorized by the TPM Owner. To use the key generation capabilities, the TPM provide a set of functions, partitioned into two classes:

- Migration functions: In this class there are many functions (*commands*) used to create and transfer migratable objects from one TPM to another for backup, upgrade or to clone a key on another platform. To do this, the TPM needs to create a data blob which holds the encrypted key exported from a TPM [21]. As an example, the TPM_CMK_CreateKey command both generates and creates a secure storage bundle for asymmetric keys whose migration is controlled by a migration authority. The resultant key must be a migratable key and can be migrated only by TPM_CMK_CreateBlob command; the command is Owner authorized via a ticket. The migrationAuth is an HMAC of the migration authority and the new key's public key, signed by tpmProof (*instead of being tpmProof*).
- Cryptographic Functions: In this class also there are many functions: as the TPM_Sign command, can be used to sign data (as a blob) and returns the resulting digital signature. The TPM_GetRandom command returns the next bytesRequested bytes from the random number generator (RNG) for any caller. The TPM_CertifyKey operation allows one key to certify the public portion of another key.

From those capabilities, we can use the TPM to create, certify and sign any keys inside itself, therefore, the TPM as a trusted and physical platform. However, in the cloud computing all physical components are shared between tenants, using the virtualization technical. Therefore, the using of the vTPM by the CKMS, consequently the inheriting of its related challenges!

V. Returned Results and Approach

From the previous section, we have captured the following main points (Fig.5):

- Using of such components (*TPM, vTPM*) as a root of trust for the CKMS as a Security Services.
- Inspiring from the architecture components and theirs deployment [6], [7], [18], [19], [20] for the CKMS solution, where these components contains the most elements of the features platform of a CKMS. This implies their interactions between them or their extension for communication with the external environment.
- Exploitation of the security solutions already proposed or improvement they for the CKMS, because we have the most same challenges with these components and our module(*migration and protection,..*).
- Creation of the trust between the customers and providers, by implementing the check of trusted chain, based on integrity measures of the CKMS instances service.

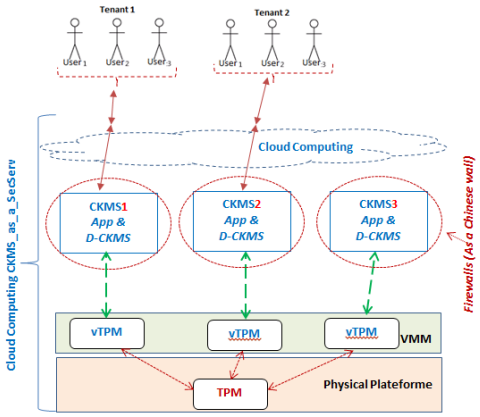

Fig. 5. CKMS as a Security Services

However, these solutions have many challenges:

- All those solutions based on the physical hardware TPM; therefore, the inheriting of the visualization problem related to migration and upgrading.
- Protection of the vTPM instances from the attacks at the VMM level; therefore the information flow control problem between the instances and from the external attacks!
- The protection of the data dictionary (*Meta-Data*) of CKMS needs others solutions as encryption to protect it.

Now, what about the benefit of TPM from the CKMS: The TPM has a limited storage capability. However, in Cloud computing there are important number of software hosted and running in the same (*shared*) physical platform, and they need important number of keys from TPM, for encryption, signing, authentication. Therefore; any enterprise key management systems/solutions (*as CKMS*) will need to:

- Support key management life cycle for TPMs
- Support the TPM as a platform identity token
- Support TPM-enabled (TPM-aware) applications
- Manage the TPM as a generic secure key store and trust anchor store

Consequently; we can resume the benefit relation between the TPM and the CKMS in the following (Fig.6):

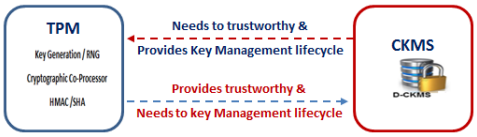

Fig. 6. TPM vs CKMS

VI. Conclusion

Outsourcing the CKMS to a trusted platform in Cloud Computing remains a real challenge. Because the security of all applications (*solutions*) based on the security of the encryption / signature keys, which are themselves dependent on the security of operation of the CKMS itself. It is within this context that reside the challenge of our research work and our contributions.

In our research work and for treating the challenges related to the trust in a multi-tenant environment (*Cloud-Computing*), we investigated in the solutions that can serve as a root of trust[6], [7], [18], [19], [20]; These solutions can ensure the software integrity, the trust chain, or using their architecture as a basis of inspiration for the development of CKMS as a Security Services.

In this paper, we have proposed an approach to provide the CKMS as a trusted security services in the cloud computing based on the TPM / vTPM. The TPM's capabilities / functions can be used as a secure way for the protection of this type of service. Therefore, the TPM used as a root of trust, and the TPM key generation component used as a secure and a trusted way to generate and to sign an CKMS's encryption/signing keys ordered by customers.

However, these solutions are based on hardware module (TPM), therefore the inheritance of the challenges related to the TPM virtualisation! Therefore, and in the future work we believe to treat the following challenge:

- Implement of the deep attestation from the CKMS's instance to the TPM (*as a physical component*).
- Using the vTPM instance as an CKMS instance, because the vTPM instance is a software component running on the VMM platform level; and this to protect the types of security as a service at this level.
- Protection of the CKMS instance by the the creation of the walls around the CKMS instance as a Chinese wall [22], and this to control the information flow between the CKMS's instance and with the other software hosted inside the same shared physical platform, or from the outside.

References

[1] P. Mell and T. Grance, "The nist definition of cloud computing," 2011. [Online]. Available: http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf

[2] A. Jerry, B. Alan, B. Alan, C. Dave, P. Nils, K. Paul, and R. Jim, "Defined categories of service 2011," *Cloud Security Alliance, Security as a service working group*, 2011. [Online]. Available: http://www.cloudsecurityalliance.org/guidance

[3] R. Janessa, "Gartner says cloud based security services market to reach 2.1 billion in 2013," Gartner, Tech. Rep., October 2013. [Online]. Available: http://www.gartner.com/newsroom/id/2616115

[4] L. Rafal, S. Dave, S. Bryan, and J. S. Luciano, "The notorious nine: cloud computing top threats in 2013," *Cloud Security Alliance, Top Threats Working Group and others*, 2013. [Online]. Available: http://www.cloudsecurityalliance.org/topthreats

[5] 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.

[6] TCG, "Tpm main part 1 design principles, specification version 1.2 revision 116," Trusted Computing Group,Copyright (c) 2003-2011 Trusted Computing Group, Incorporated, Tech. Rep., March 2011.

[7] TCG, "Virtualized trusted platform architecture specification version 1.0.26," Trusted Computing Group,Copyright (c) 2003-2011 Trusted Computing Group, Incorporated, Tech. Rep., September 27 2011.

[8] E. Barker, M. Smid, D. Branstad, and S. Chokhani, "A framework for designing cryptographic key management systems, special publication 800-130," U.S. Department of Commerce, National Institute of Standards and Technology (NIST), Tech. Rep., April 2012.

[9] C. Frederick and C. Gianpaolo, "Architecture strategies for catching the long tail, application architecture software-as-a-service (saas)," Microsoft Corporation, Tech. Rep., April 2006. [Online]. Available: https://msdn.microsoft.com/en-us/library/aa479069.aspx

[10] S. Fehis, O. Nouali, and S. Bentayeb, "Meta-data's protection in ckms-as-a-security services," in *Proceedings 4th International Conference on Information Systems and Technologies Conference ICIST 2014*, 22-24 March 2014, Valencia, Spain, pp. 195–206.

[11] TCG, "Tcg specification architecture overview, tcg specification revision 1.4," Trusted Computing Group,Copyright (c) 2003 Trusted Computing Group, Incorporated, Tech. Rep., August 2007.

[12] S. Pearson, "Trusted computing platforms, the next security solution," HP Laboratories Bristol, Tech. Rep. HPL-2002-22, November 2002.

[13] B. Balacheff, S. Pearson, L. Chen, D. Plaquin, and G. Proudler, *Trusted computing platforms: TCPA technology in context*. Prentice Hall Professional, 2003.

[14] R. Anderson, "Cryptography and competition policy: issues with'trusted computing'," in *Proceedings of the twenty-second annual symposium on Principles of distributed computing*. ACM, 2003, pp. 3–10.

[15] R. Anderson, *Security Engineering - a Guide to Building Dependable Distributed Systems*. Wiley, 2001.

[16] A.-R. Sadeghi, C. Stüble, and M. Winandy, *Information Security: 11th International Conference, ISC 2008, Taipei, Taiwan, September 15-18, 2008. Proceedings*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, ch. Property-Based TPM Virtualization, pp. 1–16.

[17] 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*. ACM, 2011, pp. 187–196.

[18] F. J. Krautheim, D. S. Phatak, and A. T. Sherman, "Introducing the trusted virtual environment module: a new mechanism for rooting trust in cloud computing," in *Trust and trustworthy computing*. Springer, 2010, pp. 211–227.

[19] D. Chang, X. Chu, Y. Qin, and D. Feng, "Tsd: a flexible root of trust for the cloud," in *Trust, Security and Privacy in Computing and Communications (TrustCom), 2012 IEEE 11th International Conference on*. IEEE, 2012, pp. 119–126.

[20] F. J. Krautheim, D. S. Phatak, and A. T. Sherman, "Private virtual infrastructure: A model for trustworthy utility cloud computing," University of Maryland Baltimore County, Baltimore, MD (2010), Tech. Rep., 2010.

[21] TCG, "Tpm main part 3 commands, specification version 1.2 level 2 revision 116," Trusted Computing Group,Copyright (c) 2003-2011 Trusted Computing Group, Incorporated, Tech. Rep., March 2011.

[22] S. Fehis, O. Nouali, and T. Kechadi, "A new chinese wall security policy model based on the subject's wall and object's wall," in *2015 First International Conference on Anti-Cybercrime (ICACC)*, Nov 2015, pp. 1–6.