DEVELOPING FRAMEWORK FOR SECURE STORAGE IN CLOUD COMPUTING SYSTEM

Rohit Bajaj
Ph D Scholar (Computer Science)
Sai Nath Universiy, Jharkhand India

Abstract - This paper shows the framework for designing the trusted platform for the cloud computing system. Data stored in third party storage systems like the cloud might not be secure since confidentiality and integrity of data are not guaranteed. Though cloud computing provides cost-effective storage services, it is a third party service and so, a client cannot trust the cloud service provider to store its data securely within the cloud. Hence, many organizations and users may not be willing to use the cloud services to store their data in the cloud until certain security guarantees are made. In this thesis, a solution to the problem of securely storing the client's data by maintaining the confidentiality and integrity of the data within the cloud is developed. Five protocols are developed which ensure that the client's data is stored only on trusted storage servers, replicated only on trusted storage servers, and guarantee that the data owners and other privileged users of that data access the data securely. The system is based on trusted computing platform technology. It uses a Trusted Platform Module, specified by the Trusted Computing Group. An encrypted file system is used to encrypt the user's data. The system provides data security against a system administrator in the cloud.

KEYWORDS: Trusted Storage, Cloud Computing, Trusted Platform, Encrypted File System, Cloud Storage

I. INTRODUCTION

Cloud Computing is the name given to a recent trend in computing service provision. This trend has seen the technological and cultural shift of computing service provision from being provided locally to being provided remotely and en masse, by third-party service providers [Hay08]. These third-parties offer consumers an affordable and exible computing service that consumers would otherwise not have been accessible, let alone afford [MKL09, Chapter 1]. This new means of service provision has evolved from and is the culmination of research stemming from (among others) distributed and networked systems, utility computing, the web and software services research [Vou08; AF+09]. This paradigm shift has led to computing being seen as another household utility, aka fifth utility", and has prompted many a business and individual to migrate parts of their IT infrastructure to the cloud and for this data to become managed and hosted by Cloud Service Providers (CSPs). However, Cloud Computing is the cause celebrate among tech pundits and has led to the term `Cloud Computing' as an umbrella term being applied to differing situations and their solutions. As such a broad range of definitions for Cloud Computing exists, each of which differ depending on the originating authors' leaning. This chapter seeks to provide a coherent and general introduction to Cloud Computing.

II. COMPUTING AS A SERVICE

One of the main tenets of Cloud Computing is the `as-a-Service' paradigm in which `some' service is offered by a Service Provider (also known as a Cloud Service Provider) to a User (consumer) for use. This service can also be categorised according to the application domain of its deployment. Examples of application domains that offer services are: Financial e.g. Mint.com, Managerial e.g. EverNote and Analytical e.g. Google Analytics. The agreed terms of use, indicating the actions that must be taken by both the provider and consumer, are described in a contract that is agreed upon before service provision. Failure to honour this agreement can lead to denial of service for the consumer or legal liability for the service provider. This contract is often described as a Terms of Service or Service Level Agreement. Moreover, as part of this agreement the service provider will provide a Privacy Policy which outlines how the users data will be stored, managed, used and protected.

III. SPI SERVICE MODEL

The services offered are often categorized using the SPI Service Model. This model represents the different layers/levels of service that can be offered to users by service providers over the different application domains and types of cloud available [MKL09, Chapter 2]. Clouds can be used to provided as-a-Service: software to use, a platform to develop on, or an infrastructure to utilise. Figure summarises the SPI Service Model. Software as a Service The first and highest layer is known as: Software as a Service (SaaS). It represents the applications that are deployed/ enabled over a cloud by CSPs. These are mature

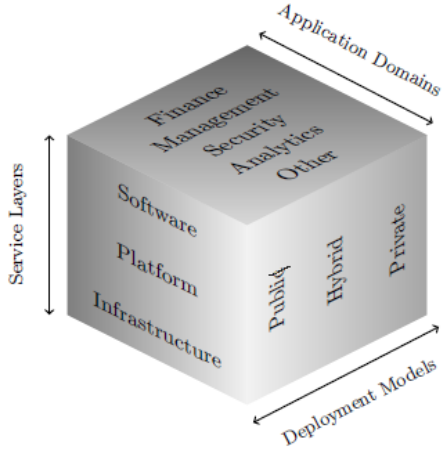

Fig 1 Summary of the SPI Service Model

applications that often offer an API to allow for greater application extensibility. For instance, Google Docs can be seen as the archetypal SaaS application, it has been deployed solely within the Cloud and offers several APIs to promote use of the application. Platform as a Service The next layer is known as: Platform as a Service (PaaS). This represents a development platform that developers can utilise to write, deploy and manage applications that run on the cloud. This can include aspects such as development, administration and management tools, run-time and data management engines, and security and user management services. For instance, Force.com and Amazon Web Services [AWS] offers a suite of services that allows developers to construct an application that is deployed using web-based tooling.

Infrastructure as a Service The final and lowest layer is known as: Infrastructure as a Service (IaaS). CSP offer developers, a highly scaled and elastic computing infrastructure that are used to run applications. This infrastructure can be comprised of virtualised servers, storage, databases and other items. Two well known examples are the Amazon Elastic Compute Cloud, a commercial platform offered as part of Amazon.com's Web Service platform and Eucalyptus, an open source platform that offers the same functionality [AWS; NW+09].

Cloud Entities

Cloud actors/entities can be divided into two main categories:

CSP or Service Provider those who provide a service;

Cloud Service User (Users) those who use a service.

Within Cloud Computing the differences between the role played by a service provider and a user can be blurred. The service provider could also be the user of another service e.g. when infrastructure is the service. The exact definition whether an entity is a provider or user is dependent on the context of the interaction and the service being offered. Some service providers will offer services at all three service levels, or offer just one particular level of service and have their own internal IaaS infrastructure.

A possible refinement could be that CSP providers are either: a) Infrastructure Service Providers|those that offer IaaS and own and run the data centers that physically house the servers and software; or b) Service Providers|those that offer PaaS or SaaS services. And that Cloud Service Users are either: A) Platform Users| are users who buy into a service providers platform e.g. Facebook; and B) Consumers|are service users who use either SaaS or IaaS services.

IV. CLOUD AND ITS TYPE

The term `cloud' has been used traditionally as a metaphor for networks and helps abstract over their inherent complexity. This term, however, has evolved to encompass the transparency between the technological infrastructure of the CSP and the consumers point of view. A cloud can be one of the following types:

Public Constituting publicly accessible services that are accessed over the Internet and are often described using the term \The Cloud".

Private These are private services deployed on private networks. Such clouds may also be managed by third parties.

Hybrid A combination of services offered both privately and publicly. For example core-services may be offered on a private cloud; other services originate from public clouds. Vaquero, Rodero-Merino et al. provides a de_nition for Clouds based upon the commonalities found among existing de_nitions [VRM+09]. They define Clouds to be: \. . . a large pool of easily usable and accessible virtualized resources (such as hard- ware, development platforms and/or services). These resources can be dynamically reconfigured to adjust to a variable load (scale), allowing also for an optimum re- source utilization. This pool of resources is typically exploited by a pay-per-use model in which guarantees are offered by the Infrastructure Provider by means of customized SLAs."

The provision of virtualised resources, as a service, allows for an elastic and on demand computing environment that can be expanded and shrunk as needed, depending on the needs of the consumer. Mather, Kumaraswamy and Latif [MKL09, Chapter 2] provides a slightly alternate definition. This definition is based on five attributes that can be used to describe a cloud-based system.

They are:

1. Multi-tenancy The sharing of resources being offered by the service providers to the consumers at the network, host and application level.
2. Massive Scalability The ability to scale the storage, bandwidth and systems available to proportions unattainable if performed by the organisation itself.
3. Elasticity Consumers can rapidly and dynamically increase and decrease the resources required as needed.
4. Pay-as-you-go Consumers only pay for the resources they consume such as processing cycles and disk space.
5. Self-Provisioning of Resources Consumers have the ability for the self-provisioning of resources.

This is a better definition, it does not pertain to a particular technology. However, one attribute Pay-as-you-go is in itself too constrictive as it pertains to a particular style of payment and prohibits subscription based models. A better attribute should be:

Flexible Payment Payment for resource usage is flexible and consumers can be offered different payment models dependent on their intended use of the resources. For instance in Amazon EC2 pricing for image use is dependent on three factors:

1. Image Type| On Demand, Reserved, Spot
2. OS Used|UNIX/LINUX or Windows; and
3. Data Center Location| West Coast America, East Coast America or Ireland. Consumers are billed differently per hour based upon these factors and other services used [AWS-Pricing]. On the other-hand Google Apps Premium Edition will cost companies $50 per user per year for their IT infrastructure solution [GooApps] and users will have a _xed set of resources. The use of such models often ensures for a lower operational cost than when compared with an in-house solution.

When discussing the physical constructions of Clouds the Data Center Model is a popular choice. This model stipulates clusters of machines running specialist, dedicated hardware and software to provide the infrastructure for the large scale provision of virtualised resources [AWS; NW+09]. Though a consumer can access the virtualised resources directly, when managing their resources, consumers interact with a Cloud Controller that mediates the interaction between the consumer, and: a) the physical machines; and b) the virtualised resources owned by the consumer. Figure 2.2 illustrates a rather simpli_ed view of this data center model. A more interesting realisation of clouds is the Adhoc model in which the existing (idle) computing power within an

enterprise is tapped to provide the required infrastructure. The more interested reader is asked to read Kirby, Dearle et al. [KD+09] for more information concerning the adhoc cloud model.

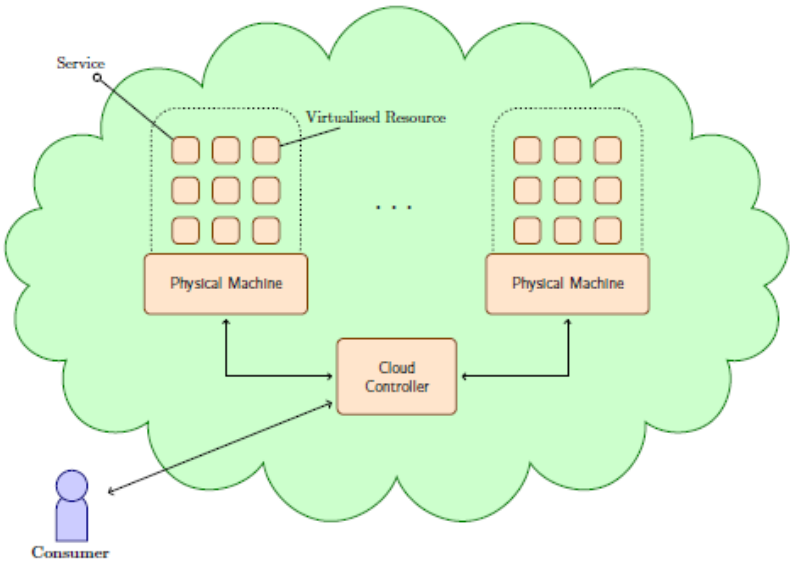

Figure 2 Data center model for cloud computing

V. MERITS OF CLOUD COMPUTING

Many of the benefits to be had when using Cloud Computing are the lower costs associated. At the infrastructure level, virtual images can be scaled and contracted with complete disregard for any associated hardware costs such as equipment procurement, storage, maintenance and use. This is all taken care of by the service provider and will be factored into the payment for the service: capital expenditure has been converted into operational expenditure. Resources within the cloud can be treated as a commodity, an `unlimited' medium. At both the platform and software level similar benefits are seen. Aspects such as software installation, deployment and maintenance is virtually non-existent. This is taken care of by the provider within their own infrastructure. The service user only pays technical support. Service providers at the SaaS level, often tout features that allow users to collaborate and interact with each other, in real-time, within the scope of the service being offered. For example, Google Docs allows users to edit documents simultaneously and for users to see each others edits in real time. Moreover, the provision of platform and software `as a service' allows cloud service users the ability to aggregate services together either for their own use or to promote as another service i.e. Mashups. The aggregation could imply the combination of functionality from several services, or the change/combination of output from the services involved. Remark. Service aggregation is a good example outlining how a service user can become a service provider.

VI. REVIEW OF LITERATUE

He and Xu [7] propose a scheme to protect data on a personal computer platform. They use the trusted computing platform developed by the trusted computing group (TCG) to develop a secure and reliable model for user authentication and data encryption. Their model uses a storage protocol to encrypt data and uses Trusted Platform Module (TPM) to authenticate different users of the PC. The host computer has the trusted computing platform installed within itself, i.e., it has TPM installed in it. First, a trusted connection is established between the host computer and the storage device, i.e., the host and the storage device authenticate each other before any data storage takes place within the storage device and before data is accessed from the storage device by the

host. Then, security control is provided by storing access control policies within the storage device. The access control policies apply to users, devices and applications. Message control between the host and the storage device is provided by implementing session-oriented secure data transmission.

Sailor, Doorn and Ward [12] describe a trusted computing platform which extends the hardware-rooted trust guarantees of TCG (Trusted Computing Group) technologies to the operating system and all its applications and allows remote parties to check the trust guarantees. The TCG defines standards for measuring and reporting integrity metrics at the system boot time. However, the TCG does not provide information on the trustworthiness of the runtime of the operating system. Sailor et al. [12] provide a solution to this problem. They give a procedure that explains how a challenging party can attest to the software stack of an untrusted machine. Trust is established between two parties after mutual attestation takes place.

The kernel of the attested system (system that needs to be attested) is instrumented to generate measurements of postboot events which affect the run-time of the system. The components of the software stack that have semantic value are measured. The measured components are kernel modules, executables and shared libraries, configuration files and other important input files that affect trust into the run-time software stack. The measurements are done as soon as executable content is loaded into the system and before it is executed. The attested system creates and stores a measurement list which is a list of hashes of the software stack components. This list is also stored in the TPM (Trusted Platform Module) for achieving integrity. The measured list and the list stored in the TPM are sent to the challenging party. The challenging party compares the two of them and trusts the measurement list of the attested system if both turn out to be the same. After that, the challenging party compares the received measurement list with its stored expected list of measurements. If the two turn out to be the same, the challenging party attests to the system and hence the challenging party is assured that the system is running the expected software stack.

Santos, Gummadi and Rodrigues [13] propose the design of a trusted cloud computing platform (TCCP) which enables the 'infrastructure as a service' providers to provide a closed box execution environment that guarantees confidential execution of a client's virtual machines. Before launching their virtual machines, the clients can attest to the service provider to determine whether or not the service provided is secure. Each node in the cloud runs a Trusted Virtual Machine Monitor (TVMM) that hosts the client's virtual machines and prevents privileged users from inspecting or modifying them. Each node in the cloud embeds a TPM to enable secure booting and for attestation. Santos et al. propose TCCP protocols to securely launch virtual machine of a client on a machine in the cloud and migrate it safely to other machines in the cloud. The TVMM guarantees that a virtual machine is launched on a trusted machine in the cloud and that the system administrator will not be able to inspect or tamper with the initial state of the virtual machine as it traverses the path between the user and the machine hosting the virtual machine.

VII. PROPOSED STORAGE SYSTEM FOR CLOUD COMPUTING

A client/user stores his data within the cloud storage servers. Since cloud is a third party system, it cannot be trusted. We consider the confidentiality and integrity issues of client's data within the cloud and provide a solution to these issues by proposing a framework/system called a Trusted Storage System for Cloud. The system provides data security against the cloud provider, especially against the system administrator who has the maximum number of privileges over the storage nodes in the cloud.

The proposed system consists of the following entities:

1. User/Client
2. Trusted Third Party Node (TTPN)
3. Front-End Server (FES)
4. Group of Storage Servers/Nodes

Role of the TPM in the system

1. Remote Attestation
2. Protected Storage of Encryption Keys
3. Secure Access of Key

VIII. JOINING A STORAGE NODE TO THE SYSTEM

A storage node in the cloud must join the Trusted Storage System before it can store a client's data. The TTPN tests if a TPM is being installed on the storage node and if installed, it checks for the correctness of the platform of the storage node This process is called attestation. After successful attestation of the platform of the storage node by the TTPN, the storage node is deemed trusted and can store a client's data securely.

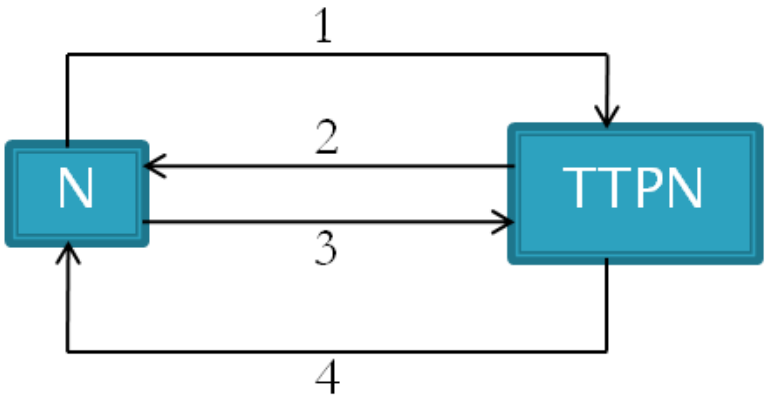

Fig : - Joining of a storage node in the Trusted Storage System of the Cloud.

IX. DATA STORAGE IN THE TRUSTED STORAGE SYSTEM

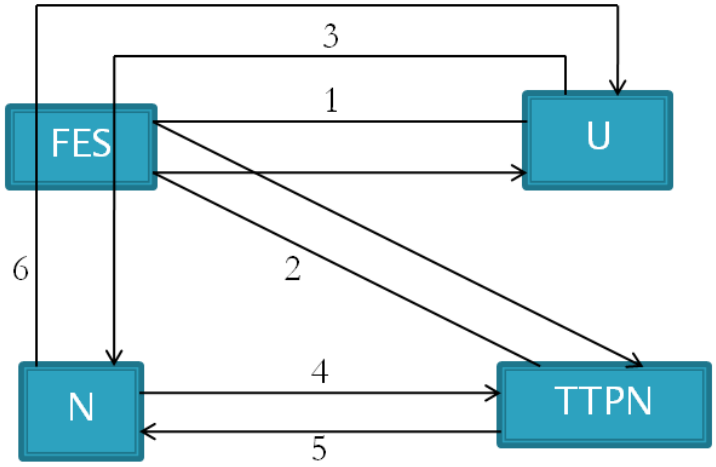

Fig:-Data Storage in the Trusted Storage System.

Protocol Used: -
Message 1: {"Data Send Req", UK ,Uid, nU } pub(TTPN)
Message 2: {{ SK, nU, nTTPN }UK, {H(nU)}pri(TTPN)}
Message 3: {{files}SK, {H({files} SK), nTTPN} pub(TTPN)}
Message 4: {Message 3, (H(Message 3))pri(N), {nN, Nid} pub(TTPN)}
Message 5: {{files}SK, {{nN, UK , Uid, SK, H({files}SK) } pub(N)} pri(TTPN)}
Message 6: {"Data Stored"}UK

The client/user sends a "data send request" (message 1) to the front-end server (FES) of the cloud to store his data in the cloud. The front-end server forwards the message to the TTPN. The user generates a symmetric encryption key, UK, called user key, to use it in further communications with the cloud. Message 1 includes the user key, user ID, Uid, and the user nonce, nU, used for preventing replay attacks. The message is encrypted with the public key of the TTPN so that only the TTPN can decrypt and view the message.

In return, the TTPN sends message 2 to the client via the front-end server. The TTPN creates a one-time session key, SK, so that the client can encrypt his data with it. This session key is used only for that user session and will not be used again. The TTPN includes its nonce in the message, nTTPN, along with the nonce of the user, nU, which it received. The TTPN encrypts the session key and the nonces with the user key UK so that only the

user can decrypt the message with UK. It signs the hash of the user's nonce, H(nU), with its private key pri(TTPN) to prove the user that the message is prepared only by the TTPN and is not tampered by an attacker.
When the user receives message 2, it decrypts the message with his key, UK, and retrieves the session key, SK, and the nonces. It also retrieves the hash of its nonce, H(nU), which is signed with the TTPN's private key, pri(TTPN). The user decrypts the hash with the TTPN's public key, pub(TTPN) and is now assured that the message is prepared by the TTPN only. The user then uses that session key to encrypt his files. He sends message 3 to the front-end server which includes the encrypted files which are being encrypted with the session key, SK. Message 3 also includes the hash of the encrypted files, H({files} SK), and the nonce of the TTPN, nTTPN, both encrypted with the public key of the TTPN, pub(TTPN).
When the front-end server receives message 3 from the client, it picks a storage server (node) within the cloud and forwards message 3 to that node. The node must authenticate itself to the TTPN before it can store the data of the client. Since the client's data is encrypted with SK, the node cannot view the client's files before it gets itself authenticated to the TTPN. The storage node N computes a hash of Message 3 and signs it with its private key, pri(N), to convey the TTPN that the message is not modified and also for its authentication. The storage node prepares Message 4 and sends it to the TTPN for authentication. Message 4 includes message 3 sent by the user, (H(Message 3))pri(N), the nonce of the node, nN, to avoid replay attacks and the ID of the storage node, Nid. The node encrypts the nonce, nN, and the node ID, Nid, with the public key of the TTPN, pub(TTPN).
When the TTPN receives message 4 from the node, it retrieves the node ID, Nid, and the nonce of the node, nN, by decrypting the content with its private key, pri(TTPN). It also retrieves message 3 and its hash, H(Message 3) signed by the node N with the private key of the node, pri(N). It checks for that node's public key, pub(N), in its database. If the TTPN finds the public key, the TTPN concludes that the storage node N is a member of the trusted storage system. The TTPN decrypts the signed hash using the node's public key. The TTPN computes a hash of the received message 3 and compares the hash with the received hash. If the hashes are the same, the TTPN is assured that the contents of message 3 are not modified. The TTPN decrypts message 3 using its private key, pri(TTPN) and retrieves its nonce, nTTPN, sent to the user in message 2 and hence considers this message 4 to be a fresh message. It also retrieves the hash of the encrypted files and the encrypted files encrypted with the one-time session key, SK. The TTPN computes a hash of the received files and compares it with the received hash. If they are not the same, the TTPN identifies that the files are modified in transit and hence does not authenticate the storage node. If the values are the same, the TTPN authenticates the node to store the user's files. If the TTPN does not find the public key of the node, pub(N), in its database, it rejects the data storage request of the node and does not authenticate the node.
After authentication, the TTPN prepares message 5 and sends it to the node. Message 5 includes the nonce of the node, nN, the user key, UK, the user ID, Uid, the session key, SK, which is used by the user to encrypt his files and also the hash of the encrypted files of the user. All of this content is encrypted using the public key of the node, pub(N), so that only the node can decrypt the content. This content is signed by the TTPN so that the node is assured that the message is prepared by the TTPN. Message 5 also contains the user's files encrypted with the session key, SK. When the storage node N receives message 5 from the TTPN, it decrypts the message using the public key of the TTPN, pub(TTPN), and now trusts that the message is prepared by TTPN only. It decrypts the inner content of the message using its private key, pri(N), and retrieves its nonce, nN, the user key, UK , the user ID, Uid, the session key, SK, and the hash of the encrypted files. The storage node also retrieves the encrypted files. It computes a hash of the received files and compares it with the received hash. If they are not the same, the node identifies that the files are modified in transit. If they are the same, the node is assured that the files are not modified. The node then decrypts the files using the session key. It uses its encrypted file system to reencrypt the files with the encrypted file system's keys. After encrypting the files, the EFS stores them in the disk. The EFS stores the corresponding encryption keys and the user key along with the user ID in the TPM. Since the encryption keys and the user key are stored in the TPM, they are safe. This is because the TPM encrypts these keys using its RSA public key whose corresponding private key is known only to the TPM.
The node now sends message 6 (encrypted with the user key, UK) to the user via the front-end server. Message 6 assures the user that his data is being stored in the cloud.

X. CONCLUSION AND RECOMMENDATION

In this paper, a framework for trusted storage of client's data within the cloud is developed. The framework provides data security against the system administrator who has the maximum number of privileges on a storage node. The system uses a Trusted Platform Module (TPM) chip in each storage node which provides protected storage of encryption/decryption keys of client's data and remote attestation of each storage node. Each storage server has an encrypted file system which encrypts the client's data and the corresponding keys are stored in the TPM. Cryptographic techniques are used to provide secure communication between the client and the cloud.

The system ensures that the client's data is stored only on trusted storage servers and it cannot be transferred by malicious system administrators to some corrupt node. The system architecture and the proposed protocols together form a Trusted Storage System for Cloud. The system achieves confidentiality and integrity of the client's data stored in the cloud.

REFERENCES

[1] Michael Miller, Cloud Computing : Web-Based Applications That Change the Way You Work and Collaborate Online. Published Aug 11, 2008 by Que. First Edition.

[2] Jian He, Mingdi Xu, Research on Storage Security Based on Trusted Computing Platform, Proceedings of the 2008 International Symposium on Electronic Commerce and Security, 2008, IEEE Computer Society

[3] Reiner Sailer, Leendert van Doorn, James P. Ward, The Role of TPM in Enterprise Security, Datenschutz und Datensicherheit (DuD), September, 2004 (also IBM Research Report 23363).

[4] Nuno Santos, Krishna P. Gummadi and Rodrigo Rodrigues, Towards Trusted Cloud Computing, Proceedings of Hot Cloud, 2009.

[5] Peter Mell and Tim Grance, The NIST Definition of Cloud Computing, Version 15, 10-7-09.

[6] T. Ristenpart, E. Tromer, H. Shacham, and S. Savage, \Hey , you , get o_ of my cloud : Exploring information leakage in third-party compute clouds," Artificial Intelligence, pp. 199{212, 2009.

[7] T. Gar_nkel, B. Pfa_, J. Chow, M. Rosenblum, and D. Boneh, \Terra: a virtual machine-based platform for trusted computing," in Proceedings of the nineteenth ACM symposium on Operating systems principles, SOSP '03, (New York, NY, USA), pp. 193{206, ACM, 2003.

[8] R. Holz, L. Lothar Braun, N. Kammenhuber, and G. Carle, \The ssl landscape { a thorough analysis of the x. 509 pki using active and passive measurements," in Internet Measurement Conference 2011, November 2-4, 2011.

[9] J. A. Halderman, S. D. Schoen, N. Heninger, W. Clarkson, W. Paul, J. A. Calandrino, A. J. Feldman, J. Appelbaum, and E. W. Felten, \Lest we remember: cold-boot attacks on encryption keys," Commun. ACM, vol. 52, pp. 91{98, May 2009.

[10] J. Schi_man, T. Moyer, H. Vijayakumar, T. Jaeger, and P. McDaniel, \Seeding clouds with trust anchors," in Proceedings of the 2010 ACM workshop on Cloud computing security workshop, CCSW '10, (New York, NY, USA), pp. 43{46, ACM, 2010.