ELG7178D
Network Security and Cryptography

Course Project

A Review of "vTPM: Virtualizing the Trusted Platform Module"

Rich Goyette
13 Nov 2007

Table of Contents

Introduction

This document provides an overview and critique of [VTPM06]. In that paper, the concepts of *virtualization* and trusted computing are merged on a single platform. The authors do this through the development of a prototype *virtual Trusted Platform Module* or **vTPM**.

The motivation behind [VTPM06] is to try to address the problem of having multiple, virtualized operating systems executing in a single hardware environment that is effectively limited to using a single *hardware Trusted Platform Module* or **hTPM**.

In this introductory section, we provide a brief overview of virtualization as well as the Trusted Computing Group's (TCG) Trusted Platform Module (TPM) specification. This is followed by a short re-statement of the problem addressed in [VTPM06] (after the required context has been provided). Finally, the organization of the remainder of this document is presented.

Virtualization

Virtualization is a new trend based upon an old idea [VEN06][MK06][WCSG05][KZB91]. Figure 1 identifies the basic concept. A typical computer with a traditional operating system can be "virtualized" by the insertion of a Virtual Machine Monitor (VMM). The VMM's task is to emulate the hardware interface seen by an operating system in such a way that the VMM can then "share" the hardware between more than one operating system at seemingly the same time. The VMM performs "context switching" between each virtual machine (VM) according to some policy that ensures each VM has reasonable access to the computing hardware.

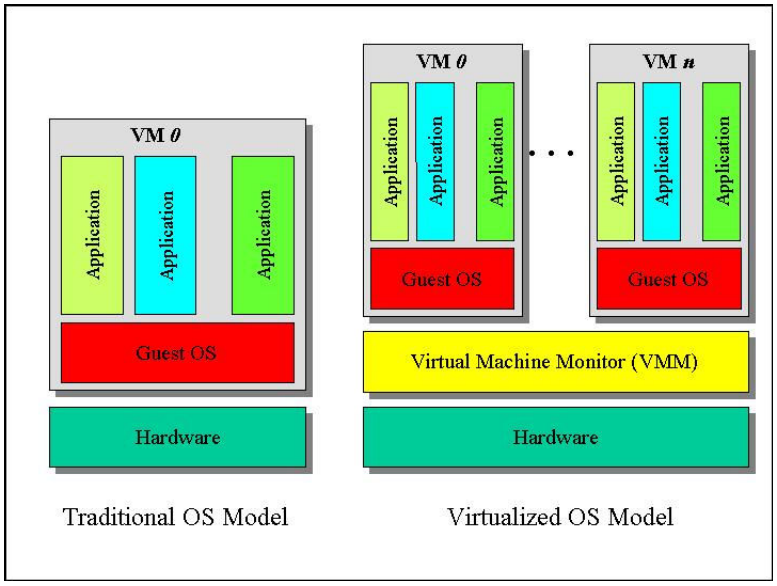


Figure 1: Basic Virtualization Concepts

Advances in virtualization have made possible a number of usage models as described in [HC05]. These are shown in Figure 2.

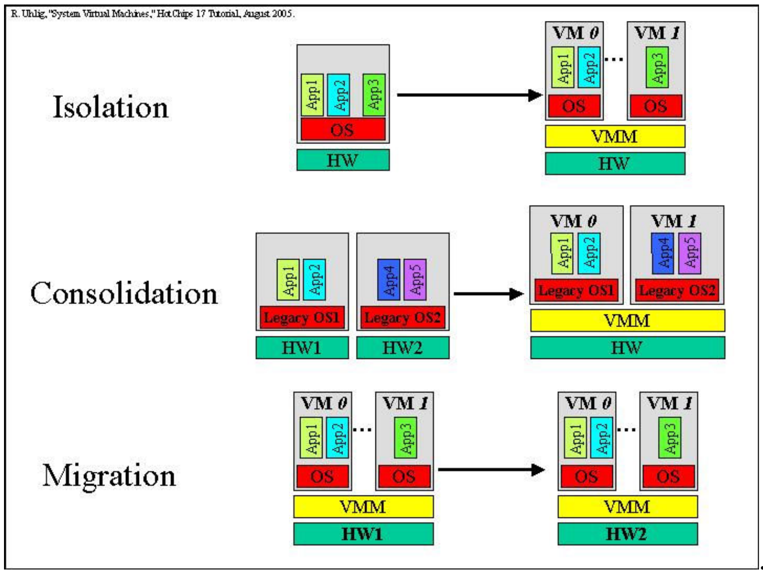


Figure 2: Virtualization Usage Models

The **isolation** usage model leverages virtualization to provide separation between various applications. This model is most appropriate within a security and availability context. Strong isolation between virtual machines can be used to secure applications processing sensitive data from each other. Strong isolation also helps prevent one application from affecting another when it crashes or when it is compromised.

The **consolidation** usage model primarily targets the reduction of hardware platforms in a workplace. Normally, consolidation wouldn't be a problem if the base operating system is consistent across an enterprise. However, in cases where critical applications exist on multiple operating systems (Linux, Windows 2000, Windows 3.1, etc), consolidation cannot occur. Virtualization side-steps the issue of adjusting business process in order to switch applications by allowing multiple legacy operating systems to execute on a common hardware base (where this is possible).

The **migration** usage model allows an arbitrary number of virtual machines to be halted, moved to a separate hardware platform, and resumed in a seamless fashion. This usage model addresses high availability requirements. When a virtualized server requires maintenance, the applications running on that server need not be stopped in order to perform the maintenance as long as a standby server is available as a migration target. This has advantages over maintaining a live backup server (not the least of which is that there is no requirement to synchronize state when the maintained server is shut down).

Trusted Platform Module

The architecture of the Trusted Platform Module (TPM) is defined by a specification that was originated (and refined over several years) by the Trusted Computing Group (TGC) [TPM07]. The TCG claims to be a non-profit, industry-standards organization whose goal is to "improve trustworthy behavior of platforms and to permit trustworthy verification" ([TNC06], p.10)[1].

In order to do this, the TCG has recognized a need for the end-user device (PC, handheld, laptop, cell-phone, etc) to provide hardware support for hiding secrets and for the evaluation of certain cryptographic functions. The TPM specification details the functionality that shall be included in a device providing this hardware support regardless of the form factor that the device takes.

Significant architectural elements of the TPM specification are described below.

Endorsement Key Pair (EK) and Certificate

Each TPM will contain a unique *Endorsement Key* (EK) pair generated for the device at time of manufacture. A certificate that binds the public portion of the EK to the TPM is generated and signed by the manufacturer and inserted into non-volatile RAM within the device. The certificate contains certain information and assertions [PIR07]:

[1] This claim won't be contested here although it should be noted that the TCG's involvement (under different names and with different member corporations) in the attempt to make "digital rights management" a mainstream technology has been a motivating factor on par with (or greater than) security. There is a rich "internet history" of these efforts starting with the Microsoft's Palladium and Intel's serial number fiascos.

a.) TPM manufacturer, model, and version;
b.) A concise descriptor of the TPM evaluation criteria;
c.) Relevant manufacturing/initialization conditions that may affect a trust decision;

Platform Configuration Registers (PCR) and "Attestation"

One of the most significant functional requirements of the TPM is to provide trusted cryptographic support for "attestation" - the process of asserting the "correctness" of an environment. For example, a certain platform P1 would like to be assured that platform P2 is running a trusted environment before it conducts any further business. An assertion of P2's correctness can be developed by "extending" one of the *Platform Configuration Registers* (PCR) in the TPM as follows:

a.) At boot time, a trusted boot-loader (BIOS) "measures" the BIOS code by performing a hash over the code to get some fixed-length value *h1*;
b.) A TPM PCR is "extended" by concatenating its current value with *h1* and then performing a hash over the result to achieve a fixed length value that fits back into the PCR;
c.) As the machine continues to boot up and invoke applications and drivers, the PCR is extended accordingly by repeating the steps above with the appropriate application binary code modules.

If P2 was booted to its current state using application binaries that are known and approved by P1 (and whose images are not altered), then the value in the PCR that measured the boot process can be predicted by P1. If P2 sends it's PCR to P1 and it is not what was predicted, then P1 can choose to stop doing business with P2.[2]

Conceptually, a business model for mutual validation would require P1 to hold one or more acceptable values for the PCR sent by P2 (perhaps to take into account the potential for different operating systems). In any case, P1 requires the ability to validate the assertion made by P2 and this means trusting a third party. The easiest way to approach this problem would be for the TPM on P2 to sign the PCR using its private EK and then send both the PCR and its EK certificate. P1 would ideally have the trusted root certificate from all TPM manufacturers and could therefore establish a third party trust in the assertion.

Attestation Identity Key-Pair (AIK)

The use of the EK private key and certificate to sign over PCR values poses a significant privacy issue because each assertion can be tied to a unique TPM. To negotiate this problem, the TPM specification calls for the ability of the TPM to produce *Attestation*

[2] Measuring or re-measuring a system that has been running for some time is an open problem. The issue is that while the measured launch process proves the system hardware and software images to be unaltered, there is difficulty in demonstrating that the currently executing image (in RAM and SWAP) is not compromised.

Identity Keys (AIK) that can be shipped off and signed by a Privacy CA in order to provide the required third-party trust. An individual AIK can be generated for each different transaction made by a person or computer and can, in theory, sever the link between an individual interacting with separate entities.

Clearly there are weaknesses in this approach (e.g. widespread availability and *trust* in a Privacy CA) and a separate effort has been specified to try to rectify the privacy issue without requiring a significant third party trust infrastructure. This approach, known as Direct Anonymous Attestation ([DAA07]) does succeed in reducing the complexity but still has significant deployment complexity.

Dichotomy Between Virtualization and TCG TPM

As indicated earlier, the aim of [VTPM06] is to try to reconcile the apparent differences between two very significant technologies. The TCG TPM promises to provide trust and security to compute platforms (including servers) but was only designed to support a single "owner" [PER06].

On the other hand, virtualization brings the potential to reduce a network's footprint by consolidating security policy domains (as well as legacy operating system domains) and increase the utilization of server resources. However, virtualization provides many prospective owners to a single TPM thereby making the act of attestation difficult.

Organization

The following section provides an analysis of three key areas of [VTPM06] that we thought could be improved. For each area, a structured approach is taken with the following sub-section headings:

a.) **Title**: a descriptive moniker;
b.) **Assertion**: a concise statement of a perceived shortcoming;
c.) **Discussion**: a description and reasoning behind the perception of the shortcoming;
d.) **Improvement**: a suggested improvement to the paper that would address the shortcoming;
e.) **Critique**: a critique of the proposed improvement.

The first area of potential improvement is in the description of the secure migration protocol. This is followed by suggestions for secure VM-vTPM association. The final area highlights the paper's dependency on privacy infrastructures. Following these discussions, we present summary and concluding statements.

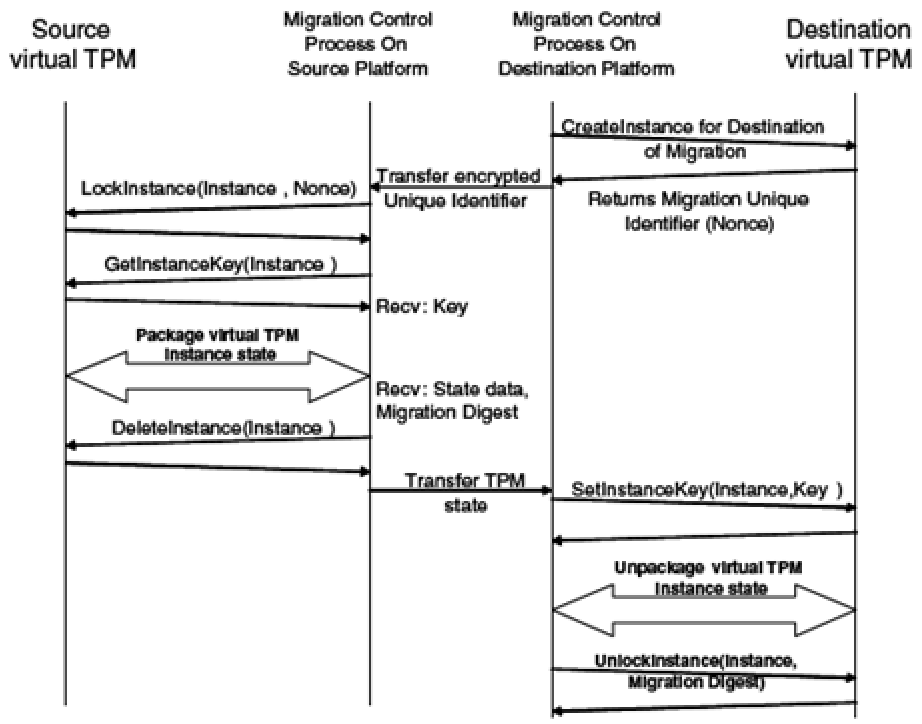
Secure Migration Control

Assertion

A considerable portion of [VTPM06] was devoted to discussing how the implementation would work *securely* under the migration usage model. However, this discussion was not entirely convincing from a security perspective because the authors failed to provide adequate background in certain key areas. Specifically, a description of the migration control process at a system level was not undertaken or referenced to a depth sufficient to understand the full security management implications.

Discussion

In Section 4.5 of [VTPM06], the key migration protocol for a vTPM is described in detail. The protocol sequence diagram (Figure 3 of [VTPM06]) is repeated in Figure 3 for discussion purposes.



Source virtual TPM
Migration Control Process On Source Platform
Migration Control Process On Destination Platform
Destination virtual TPM
CreateInstance for Destination of Migration
Transfer encrypted Unique Identifier
Returns Migration Unique Identifier (Nonce)
LockInstance(Instance , Nonce)
GetInstanceKey(Instance)
Recv: Key
Package virtual TPM Instance state
Recv: State data, Migration Digest
DeleteInstance(Instance)
Transfer TPM state
SetInstanceKey(Instance,Key)
Unpackage virtual TPM Instance state
UnlockInstance(Instance, Migration Digest)

Figure 3: vTPM Migration Protocol (Figure 3 of [VTPM06])

One of the first concerns that becomes obvious in reviewing the migration protocol was that it did not describe how the key(s) that are used to encrypt a migrating vTPM are securely moved between platforms. Figure 3 indicates that the "migration control process" at the source receives a key from a (parent) vTPM and that this key is used at the destination to decrypt the migrated vTPM into a new state-space. The paper does not describe this control process but rather indicates that it is based on "TPM key migration" ([VTPM06], p. 316). In fact, the extent to which this process is described is identified in Figure 3 where the two inner sequence points are identified simply as "Migration Control Process."

A literature search reveals that there are currently only two TCG-supported "protocols" for migrating material between hardware TPMs [RH107]. These include *immediate migration* (identified within the TPM specification [TPM05] as "rewrap") and a more robust, enterprise-based *migration infrastructure* (identified by the TCG Interoperability Specification for Backup and Migration [ISBM05]). It was not entirely clear on first (and second) reading which one is used in [VTPM06] but this was irrelevant because neither demonstrates a reasonable solution to the problem of trustworthy migration.

Immediate migration is a protocol for *secure key transfer* (rewrap) and is supported in TPM v1.1 and v1.2. However, this protocol is limited specifically to the packaging of *keys* for *transfer*. Other important aspects of a trustworthy migration environment are left to the application developer to address. These include (but are certainly not limited to) ensuring that the destination TPM is a trusted platform, restricting or otherwise logging the number of duplicate keys made [RH107], ensuring that a "man-in-the-middle" attack does not occur during key transfer, and the extremely important questions surrounding management and scalability.

In the transfer protocol of Figure 3, an attempt is made to "shore-up" some of these missing trust elements by specifying, for example, the requirement for the destination vTPM to generate, encrypt and transfer a unique identifier to the source in order to prevent duplicate copies of the migrating TPM state. However, the transaction "Transfer encrypted unique identifier" is never described in the paper so it is not clear what keys are being used for this encryption. If this key is symmetric, then there must be a management step in the protocol to implement provisioning of the key (securely) with each parent vTPM, management of those secret keys throughout the lifecycle of many vTPMs across many computers, and automated procedures for key compromise. Clearly, this is non-trivial to achieve in a complex environment. On the other hand, if the encryption key is a public key for the source vTPM, then there must be a step in the protocol that allows transfer (or fetching) of the key prior to encryption. To prevent "man-in-the-middle" attacks, this key needs to be part of a certificate signed by a trusted third party (since the EK cannot sign over certificates). In other words, there must be an "infrastructure" to support this public key exchange.

The second method for migration resulted from the work of the Data Migration Task Force (DMTF) as part of the Infrastructure Working Group (IWG) within the TCG. It is based on the TPM v1.2b specification and introduces the concepts of Certified Migration Keys (CMK) and Migration Authorities (MA). These artifacts invoke complexity on an order roughly equivalent to a public key infrastructure and there is no evidence in the paper to support that this migration approach was taken. Of note, at least one claim is made that no DMTF specification-based infrastructure currently exists to support key migration using CMK or MA [WIN07]. A (non-exhaustive) survey of available commercial offerings appears to support this claim.

Proposed Improvement

Obviously, if all vTPMs are allowed to migrate and each machine in an organization could host several tens of VMs, then a migration control process will be required to handle the complexity and it suddenly becomes a key security enabler.

The migration protocol section of this paper could be improved considerably by the provision of some additional information and a slight modification in the way the explanation of migration control was carried out. Specifically, a top-down description starting with a deeper exploration of currently implemented means of TPM migration (and more importantly their limitations and associated overhead) would allay immediate concerns of security-oriented readers. Failing this, a clearer identification of which TPM migration tools were being relied upon could be made followed by curt direction to the appropriate references.

If a detailed description was provided, it would have to contain more depth than what was presented. Specifically, a diagram similar to that of Figure 3 would be needed to identify, in a quick fashion, the protocols supported for TPM migration, the complexities and overheads involved, and identify which features were being used by the vTPM migration protocol.

Critique

After several readings and some not-insignificant background research, a novice reviewer (i.e. someone who is not conversant with the TPM or its structures) could understand the migration approach and make an educated guess at the implicit complexities - those elements that were "assumed away" to allow the authors of the paper to get to some point. However, it is quite common for authors of technical publications to target their audience. The inclusion of additional detail on what is assumed to be known can take away from other areas that are considerably more important for the knowledgeable reviewer in a space constrained journal.

Indeed, an extended discussion of the migration control process at large might distract the reviewer (both novice and seasoned) in an already lengthy and difficult-to-follow paper.

Secure VM-vTPM Association

Assertion

The proof of concept presented in [VTPM06] was designed to meet several security critical requirements. One of these was that each vTPM is associated strongly with a unique VM. The evidence provided to substantiate this claim was not convincing.

Discussion

The paper makes a claim of a "strong virtual machine to virtual TPM association." where the word *strong* is used in the sense of *secure*. The security of this binding is based on a split driver design in which the "head" (or server-side) portion of the driver exists in the vTPM manager domain and the "tail" (or client-side) exists in the VM as shown in Figure 4 (slide 39 from [PER06]).

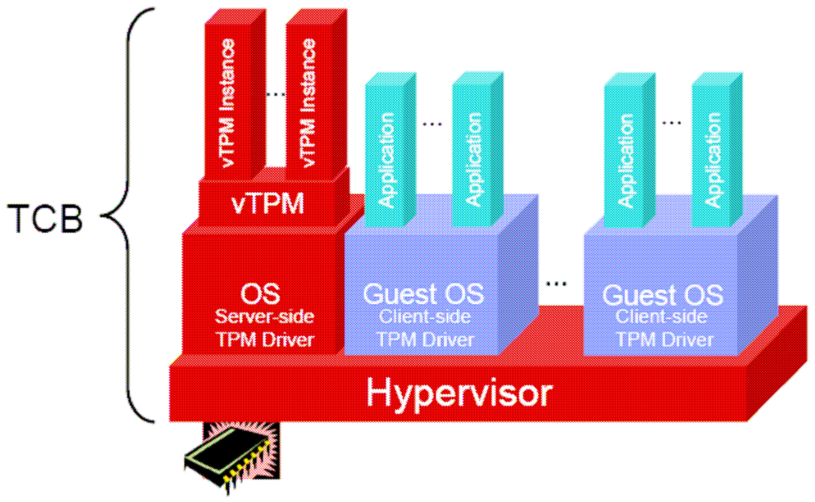


Figure 4: vTPM Trusted Computing Base (TCB) (slide 39 from [PER06]).

All of the vTPMs are concentrated into a single virtual machine that forms part of the Trusted Computing Base (TCB). Access to these vTPMs is mediated by a vTPM manager. The decision to place the vTPMs into a single virtual machine was taken by the authors to achieve a related security requirement - establishing a strong association between each vTPM and the underlying TCB.

With this architecture, the authors point out that it is not possible for a vTPM to know which VM originated a TPM command. Therefore, the vTPM manager attaches a four-bit address code to each VM-vTPM transmission to establish a mapping between them and then maintains this table throughout the lifecycle of all VMs. It is this mechanism that is the basis of the claim that the proof-of-concept has a strong VM to vTPM association. Since the address code is appended within the TCB, then it should not be possible for a VM to address a vTPM that it does not own.

However, the client-side driver does not belong to the TCB since it exists within a guest operating system. Although it is possible to take a measurement of the guest driver on VM launch, taking a meaningful post-launch measurement is still very much a challenge because binaries exist in execution space and contain significantly more variable data. It is therefore possible for the client-side driver to be compromised without detection. And, since this driver uses an *event* to communicate to the server-side driver in the proof-of-concept, it is not inconceivable for a compromised client-side driver to address *any* arbitrary vTPM by generating the appropriate event signal.

In the case of a compromise as described above, the TCB is still trusted. However, it is simply performing its duties faithfully on behalf of a misbehaving VM. And, because all of the vTPMs exist within a single managed VM, all of them are subject to subversion. This is an example of the dangers inherent in breaking the isolation principle of VMs and allowing inter-VM communication.

Proposed Improvement

Clearly, the placement of the code that implements the vTPM is critical to both its trustworthiness and to the amount of effort that must be invested in attesting to its correct operation. If the vTPM is placed outside of the VM to which it is bound (as in this case), then correct operation can be "asserted" (by assuming it belongs to the TCB) but at the expense of requiring strong identification of any VM making a request to a vTPM. On the other hand, if the vTPM is placed within the VM to which it is bound, isolation is guaranteed (solving the identification problem) but the problem of asserting correct, continuous vTPM operation is then encountered.

From a security perspective, there are significant advantages to be gained by maximizing isolation between virtual machines. This argues strongly in favour of implementing vTPMs within the VM to which they are bound or in separate, dedicated VMs (one TPM VM for each new VM). This forms the nucleus of the proposed improvement. The advantages of either approach are that Inter-VM communications are eliminated or minimized (cutting off these vectors of attack), complexity is reduced (no vTPM management domain is required) and the size of the TCB can be reduced

Critique

The question of whether a vTPM is behaving "correctly" is currently difficult to answer when it is implemented as part of a VM because the target OS is not part of the TCB (as was assumed in [VTPM06]). As noted in [AAT07], a minimum of three entities are needed to perform an attestation: a **Target**, an **Attestor**, and an **Appraisor**. To be credible, an attestation must be made by a trustworthy, independent mechanism (the Attestor) that exists in its own *security domain* and is able to securely communicate with a potentially compromised VM (the Target). From that viewpoint, the Attestor must be part of either the VMM (which should be avoided) or exist in its own VM. The authors of [AAT07] argue that the Attestor must be small enough to be assured but flexible enough to provide the functionality needed for attestation services but none of the current architectures (virtual machines, security kernels) achieve these properties at this time

Another issue with having the vTPM within the target VM (or its own VM) is that of the "freshness" of a measurement. A VM can be compromised at any time after initial startup (when the first measurement is taken). This leaves the open question of when and how to re-measure those parts of the OS that are related to the vTPM. Should the vTPM be re-measured every time a vTPM transaction occurs or at some less frequent interval? How is the dynamic portion of the vTPM code measured and re-measured? As noted in [AAT07], the concept of "measurement" as simply a series of hash values needs to be revisited in order to provide additional flexibility in gathering assurance evidence.

A final critique involves performance. There is a cost to each context switch and this will inevitably place an upper bound on the number of VMs that can be instantiated. With all the vTPMs concentrated in a single management VM, the additional cost of virtualizing the architecture is one more VM. Placing the vTPM within the guest OS offers no additional cost to virtualization but, as we hinted earlier, may consume equal or more bandwidth in post-launch re-measurement and validation. The final option would see each guest OS receive a dedicated VM containing only the vTPM and a basic operating system. Clearly, this is the worst-case virtualization cost since the number of VMs must double to support virtualizing the TPM.

Dependency on Privacy Infrastructures

Assertion

The paper proposes a number of solutions to establish a transitive trust link between the EK generated by the vTPM and those issued with the hTPM. These solutions attempt to work within the constraints of the current TPM specification and make use of privacy structures to solve the transitive trust problem. This results in a dependency on significant infrastructure which may not always be necessary to achieve the goals of the TCG - especially when privacy is not a concern.

Discussion

One of the problems tackled in [VTPM06] was to attempt to address the generation of certificates for the EK of vTPMs (denoted here as EK'). This certificate is used as part of the evidence provided by a TPM to obtain an Attestation Identity Key (AIK). As discussed briefly at the outset, AIKs were implemented by the TCG as part of the TPM Version 1.1 specification in order to speak to a privacy issue. Specifically, it was realized that if an hTPM-unique EK was used to sign attestation requests, then it would be possible to track an individual on a network as she moved from location to location. The solution was to give the hTPM the ability to generate AIK that would be signed by a Privacy CA according to the protocol shown in Figure 5 (a simplified version of slide 38 of [WIN07]) and these would be used in lieu of the hTPM EK.

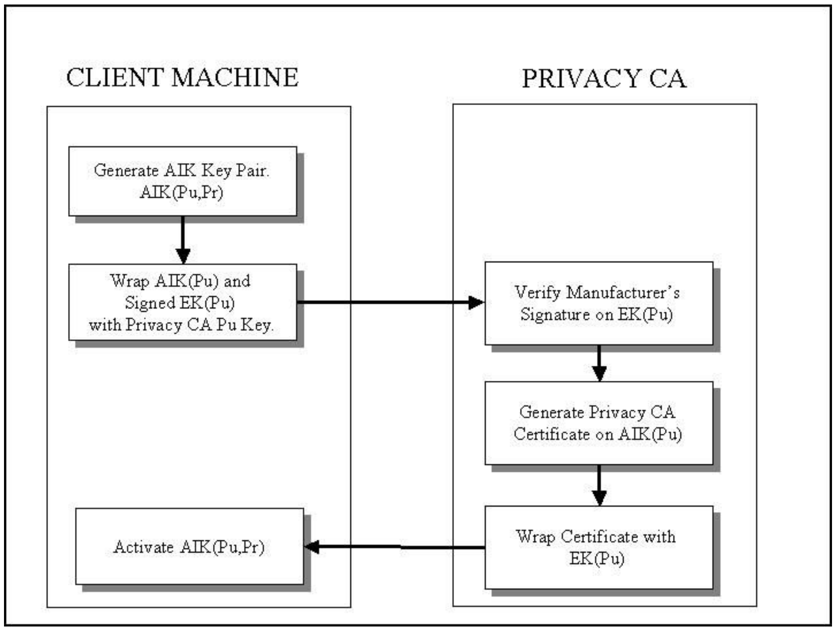

CLIENT MACHINE
PRIVACY CA
Generate AIK Key Pair. AIK(Pu,Pr)
Wrap AIK(Pu) and Signed EK(Pu) with Privacy CA Pu Key.
Verify Manufacturer's Signature on EK(Pu)
Generate Privacy CA Certificate on AIK(Pu)
Wrap Certificate with EK(Pu)
Activate AIK(Pu,Pr)

Figure 5: AIK-Privacy CA Protocol (Slide 38 from [WIN07])

In Figure 5, the Client Machine generates an AIK key pair within the TPM and wraps them with the public key provided by a Privacy CA. Included in this "bundle" is the public portion of the TPM's EK in a certificate signed by the manufacturer. The Privacy CA verifies this signature using a local copy of the manufacturers trusted root certificate and determines if the platform originating the request is trustworthy. If so, it generates and signs over a certificate for the public portion of the AIK that was submitted. It wraps this certificate with the public EK (found in the TPM certificate signed by the manufacturer) and sends it back to the requesting client.

The Client Machine can now use the signed AIK to sign over PCR values that some third party will require in order to validate the "correctness" of the Client's state. The third part must have access to the Privacy CA's trusted root certificate instead of all potential manufacturer's certificates.

The transactions shown in Figure 5 should be repeatable with a virtual TPM. However, the current specification prohibits the TPM from signing *anything* with the private portion of the EK in order to guarantee privacy. Therefore, other means are required to convince a Privacy CA or other *verifier* that a virtual TPM is related to a hardware TPM. The authors propose the following three approaches:

a. *vTPM EK' to hTPM AIK Binding*: In this approach, the hTPM creates an AIK and has it signed by a privacy CA. The hTPM uses this AIK to sign over the vTPM EK certificate. To provide a binding to the hardware TPM, an "attestation" is performed using the a hash of the vTPM EK certificate and hardware PCRs and this is inserted back into the vTPM EK certificate.

b. *EK Certificate Signing Ca*: In this approach, a separate CA instance (in addition to the Privacy CA) is invoked specifically to provide signatures over the EK of each new vTPM. Note that this approach does not provide any connection between the vTPM and the hardware platform on which it exists.

c. hTPM AIK signs vTPM AIK': In this approach, the EK' for each vTPM is ignored. AIK' generated by a vTPM are signed by an AIK generated by the hTPM. As indicated in [VTPM06], this breaks the protocol for obtaining signatures across an AIK.

A fourth solution replaces the TPM altogether with a secure coprocessor. The implication in [VTPM06] is that the EK used in the secure coprocessor can be linked to the endorsement key for the secure co-processor itself making all of the options above valid alternatives.

All of the solutions bind a new vTPM to the hTPM via AIK (either directly or indirectly). A more "natural" approach would have required the hTPM EK to sign over a certificate generated for each new vTPM EK'. In this case, a verifier - whether it is a Privacy CA or some other system element, needs only to have the original manufacturer's self-signed root certificate to process the trust chain. However, as discussed earlier, the TPM EK is prohibited by the specification from signing to absolutely preserve privacy.

The by-product of this approach is that the solutions proposed *must* incorporate either a privacy CA or Direct Anonymous Attestation (DAA) infrastructure regardless of whether privacy is a concern or not. DAA is the TPM Version 1.2 approach to eliminating the need for a Privacy CA because of inherent limitations of the Privacy CA concept ([DAA07] [RH107]). These limitations include the fact that a Privacy CA has to be both highly available (to sign every new AIK) yet highly secure (which is difficult to achieve in practice) as well as the fact that a Privacy CA could willingly provide privacy information to retailers or law enforcement without a user's knowledge or consent. DAA, on the other hand, is a method of generating AIKs without giving away privacy information or seeking the assistance of a Privacy CA. However, this method also requires infrastructure support that is not provided by traditional asymmetric key building blocks.

Proposed Improvement

The paper could be extended by the inclusion of a fourth option that would address changing the specification for the TPM to allow EK signing and a discussion of the resulting changes to virtual TPM trust chaining that result.

Figure 6 shows how the process of directly signing vTPM EK' would work and identifies its major implications for implementation.

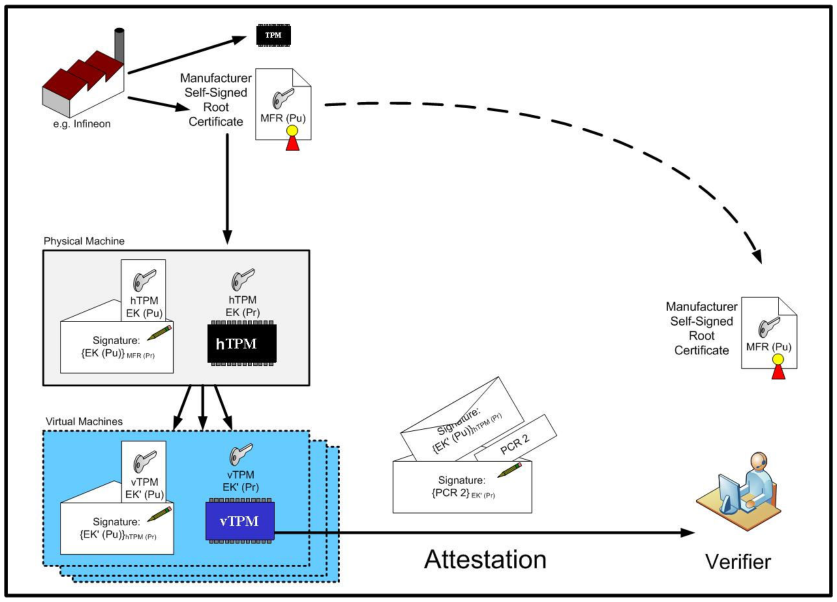

TPM
e.g. Infineon
Manufacturer Self-Signed Root Certificate
MFR (Pu)
Physical Machine
hTPM EK (Pu)
Signature: {EK (Pu)} MFR (Pr)
hTPM EK (Pr)
hTPM
Virtual Machines
vTPM EK' (Pu)
Signature: {EK' (Pu)}hTPM (Pr)
vTPM EK' (Pr)
vTPM
Signature: {EK' (Pu)}hTPM (Pr)
PCR 2
Signature: {PCR 2} EK' (Pr)
Manufacturer Self-Signed Root Certificate
MFR (Pu)
Attestation
Verifier

Figure 6: Attestation Process with EK Signing

As shown in Figure 6, transitive trust is established all the way from the manufacturer of the TPM down to whatever nested level of vTPM is required. The hardware TPM (hTPM) has been loaded with a certificate containing its public key that has been signed by the manufacturer (e.g. Infineon) or produced locally. As each virtual machine is created, the public portion of the Endorsement Key for each virtual TPM (vTPM) is signed by the private key of the hTPM as part of a certificate signature.

When it is time to make an attestation, the vTPM can still make use of one of the existing AIK protocols for privacy (e.g. Privacy CA, DAA) whenever implementations of those protocols become sufficiently scalable and supportable as to be useful. In the meantime, a virtual TPM can make a very strongly bound attestation directly as shown in Figure 6 without the overhead required for the privacy concerns. The verifier needs only have the manufacturer's trusted root certificate to complete the verification of the chain of trust.

Compare this to the worst-case scenario (option b. above) where each new vTPM must securely transact with an EK' CA and each new AIK' generated by the vTPM must interact securely with a Privacy CA (through the hTPM) before the AIK' can be used. In addition to the disadvantages of a Privacy CA noted previously, there is the problem of establishing a CA within an organization (not to mention two of the them). A CA is more than a network identity service - it invokes a massive overhead in policy, procedures, and personnel in order to do securely and correctly.

As noted, the proposed improvement would not significantly change the outcome of the overall paper because a Privacy CA or DAA is still required *as long as privacy is a concern*. However, the extension proposed here allows the paper to be scalable to problem spaces where privacy *is not at issue*. It can be argued that this includes almost all current and future applications where strong network security and integrity is desired. In these situations, privacy is problematic because strong identification and authentication is generally a precondition for network elements or individuals to access the network or network resources.

For example, Trusted Network Connect (TNC) is a Trusted Computing Group (TCG) specification that allows a *verifier* on a server to make decisions on extending the enterprise boundary to a platform that has requested connectivity based on " the *integrity information* reported by the platform and by the *proof-of-identity* supplied by the platform.' ([TNC06].p.10). This is a classic example of securing trust that the endpoint has booted in a trusted fashion, contains only approved, unmodified applications, has a complete virus database, belongs to the class of permitted hosts, etc. There are policy applications available to do this today but they are not rooted in a hardware-based trust anchor and are therefore more susceptible to compromise than a TPM-based approach. The disadvantage of the TNC approach is that the current specification binds identities to non-migrateble AIK thus *requiring* either a Privacy CA or DAA in order to establish a trusted connection. If TPM EK signing was allowed, this specification would also need to be revisited.

Critique

The question of when and under what conditions the EK could sign a document becomes extremely relevant when it is suddenly allowed to do so. For example, if it is possible for the EK to sign something legitimately, could it also be possible to coerce the TPM into signing something that was not intended? Such an act could expose a user's identification in a privacy scenario or, worse, force a false attribution upon the user.

These considerations are no doubt something that have been considered by the TCG in the past and have probably shaped the decision to prevent TPM EK signing. To propose that the TPM EK be allowed to sign without addressing these concerns only adds to the problem. However, to do so would clearly be out of scope of the original paper.

As a final note, it has been argued that TPM EK signing would allow network security and endpoint integrity verification to proceed without the need for a Privacy CA and/or DAA infrastructure. This is not entirely the case since the current generation of TPMs do not natively support TPM EK signing. A period of time would be required before specification changes made their way into hardware TPMs. The question is, which will arrive first - a scalable, acceptable privacy infrastructure or TPMs capable of using their EK for signatures.

Summary and Conclusions

In this document, a review of [VTPM06] was performed and several areas of potential improvement were identified. It was perceived that the description of secure migration fell somewhat short of identifying the entire complexity of the problem at hand since the infrastructure requirements were not disclosed or discussed. A suggestion was made to extend the paper with an enumeration of the migration protocols currently supported by the TCG and an assessment of technology maturity along those specifications.

The assertion in [VTPM06] that there was a strong binding between each vTPM and its associated VM was found to lack credibility due to its reliance on drivers existing in the non-TCB guest operating system. Because all vTPM were concentrated into a single, managed VM, this binding failure could lead to any VM accessing any other vTPM. It was proposed that the vTPM be distributed either to each guest OS or, more appropriately, to individual VMs bound pair-wise to each guest VM.

Finally, it was noted that the trust chaining solutions provided in [VTPM06] to link virtual EK to hardware EK were dependent on privacy mechanisms (Privacy CA or DAA) or did not link the EK directly to the hardware (i.e. made use of a separate CA). It was suggested that the range of solutions could be expanded by making a case to have the specification for the TPM and related TCG components revisited to make it possible for the EK of a TPM to sign documents. The benefit would be to enable the range of solutions that seek trust and integrity on a network where privacy is not a concern.

References

[AAT07] Sheehy, G. Coker, J. Guttman, P. Loscocco, A. Herzog, J. Millen, L. Monk, J. Ramsdell, B. Sniffen, "Attestation: Evidence and Trust," Mitre Technical Paper, March 2007. Online at http://www.mitre.org/work/tech_papers/tech_papers_07/07_0186/

[DAA07] IBM Direct Anonymous Attestation. Online at: http://www.zurich.ibm.com/security/daa/

[HC05] Uhlig, R., "System Virtual Machines," HotChips 17 Tutorial, August 2005.

[ISBM05] Trusted Computing Group Interoperability Specification for Backup and Migration, Version 1.0, Revision 1.0, 2005.

[KZB91] Karger, M. Zurko, D. Bonin, A. Mason, C. Kahn, "A Retrospective on the VAX VMM Security Kernel," IEEE Trans. on Software Engineering, Vol 17, No. 11, Nov 1991, pp. 1147-1165.

[MK06] Meier, T. Kockler, "Many in One," Linux Magazine, Issue 70, Sept 2006.

[PER06] Perez, "Virtualization and the Trusted Platform Module," Presentation Slides, Second Workshop on Advances in Trusted Computing, Dec 2006. Online at: http://www.trl.ibm.com/projects/watc/program.htm

[PIR07] Pirker, "Trusted Computing Infrastructure," Lecture Presentation Slides, Trusted Computing Labs, IAIK TU Graz. Online at: www.iaik.tugraz.at/teaching/04_**trustedcomputing**/slides/pki.pdf

[RH107] Gallery E., "Trusted Computing IY5608: The RTM and TPM Infrastructure Working Group Activity," Course Lecture Notes, 16 Feb 2007, website, accessed on 10 Oct 2007.

[TCB07] McCune, B. Parno, A. Perrig, M. Reiter, A. Seshadri, "Minimal TCB Code Execution," *Proceedings of the 2007 IEEE Symposium on Security and Privacy*, May 2007,pp. 267–272.

[TNC06] Trusted Computing Group. TCG Trusted Network Connect: TNC Architecture for Interoperability,May 2006. Version 1.1

[TPM07] TPM Main Part 2 TPM Structures, Specification Version 1.2, Level 2, Revision 103, 9 July 2007, Trusted Computing Group.

[VEN06] Venezia, P., "Vitualization Breaks Out," InfoWorld, Issue 27, 3 July 2006.

[VTPM06] Berger, R. Caceres, K. Goldman, R. Perez, R. Sailer, L. van Doorn, "vTPM: Virtualizing the Trusted Platform Module," 15th USENIX Security Symposium, 31 July - 4 Aug, 2006.

[WCSG05] Whitaker, R. Cox, M. Shaw, S. Gribble, "Rethinking the Design of Virtual Machine Monitors", *IEEE Computer,* Vol. 38, Issue 5, May 2005, pp. 57-62.

[WIN07] Winkler, "TPM-Trusted Platform Module", Presentation Slides, Trusted Computing Labs, IAIK TU Graz, March 2007. Online at: