An Implementer's Guide to Establishing vTPM Trust

Ariel Segall

May 11, 2010

1 About This Document

In this document, we set out an outline of how to create a trustworthy vTPM architecture. We will not cover full details of how this architecture results in long-term, certifiable trust of vTPMs, but will sketch out the reasoning and include detailed lists of which properties are necessary for which parts of a vTPM's lifecycle.

In each lifecycle stage, we will cover the steps that are essential to preserving the long-term trust of a vTPM. We will also call out areas of variation, where the techniques we use are not the only trustworthy option; in these cases, implementation variations are not only possible but expected, as long as the necessary properties are preserved.

2 Assumptions

We make the following assumptions about the system, which are necessary to establish trust. An implementation should, as much as is possible, support these assumptions.

- Boot-time measurement hashes in TPM PCRs or of VM images are predictive of future software behavior. In other words, if a measurement of a component is correct, we can assume that the component will perform as expected. (Note: In order to trust a vTPM, this assumption only actually needs to hold true for the vTPM, vTPM Manager, and other components providing the assumptions listed here.)
- The virtualization environment provides reliable separation between components, particularly between vTPMs and other components. Another component running on the same system as a vTPM should not be able to access the vTPM's internal memory or state, regardless of what the component is. This includes other vTPMs, the vTPM's associated user OS domain, and the vTPM manager.

- A mechanism for providing secrecy and authenticity of communications between system components (particularly in separate VMs) exists. In particular, traffic between vTPMs and the vTPM manager should not be readable or modifiable by an adversary, and the vTPM Manager should be capable of confirming that particular messages came from a given vTPM without additional cryptographic confirmation.

- The vTPM Manager is the only component with direct access to the hardware TPM; any other components attempting to use the hwTPM must be approved by the vTPM Manager.

- We refer to the various system components which support our assumptions as the *trusted base*. At a bare minimum, the trusted base includes the hypervisor and its MAC policy. All trusted base components should be measured into the hardware TPM. Trusted base components include the vTPM Manager; the Domain Builder or other component responsible for correctly creating and measuring domains; the Control Domain or other component responsible for correctly associating domains with each other as part of virtual platforms; and the hypervisor or other component responsible for enforcing separation and MAC over the hard drive and VMs.

3 Adversary Assumptions

- We assume that an adversary can freely access the machine's hard drive between "good" boots. The adversary can also write over any data on the hard drive.

- We assume that an adversary can intercept any non-protected traffic between VMs, and can freely intercept network traffic.

- We assume that the adversary can access the TPM during "bad" boot cycles, but does not have owner authorization. [1]

- We assume that the adversary cannot violate the TPM's behavioral guarantees, e.g. by performing hardware modification.

4 Summary

When we discuss trust in a vTPM, we mean two things: we want a vTPM to behave in a manner that is comparable to a hardware TPM, and we want the vTPM's secrets to only be accessible to the vTPM itself. We wish to establish

[1] An adversary with physical access can perform a DoS on the system, and render all existing vTPMs inaccessible. This parallels the owner takeover attack on a hardware TPM, which renders all existing keys inaccessible. Our concern in this document, however, is preventing the use of a vTPM key by a non-vTPM component or "bad" vTPM.

long-term remote trust in a vTPM: we want a remote party to be able to receive some chain of trust, in our case in the form of certificates, rooted in a trusted authority, and derive from this chain that a particular secret belongs only to a trustworthy vTPM. Trust in this secret (the Root of Trust for Reporting) and the vTPM can be used to establish trust in other keys or system components.

At a high level, our argument for long-term trust in a vTPM goes as follows:

- A vTPM that booted correctly will always shut down safely, preserving all of its data so that it can only be accessed under the correct circumstances.
- A vTPM that shuts down safely will always either boot correctly the next time it boots, or be unable to boot.[2]
- A vTPM manager can certify that a vTPM has booted correctly once, and therefore that it will always boot correctly in the future.

A running vTPM is responsible for ensuring that any long-term changes to its data are preserved immediately, and that all secret data will only be available to itself, running on a good machine. We ensure this by using the hardware TPM to make all vTPM data only accessible when certain PCR values are present.

The vTPM manager is responsible for ensuring that only one intance of a given vTPM is running at a time, and that only the most recent vTPM state is available, thus ensuring that the virtual nature of the vTPM does not allow us to duplicate the trusted "hardware" or roll back time. The vTPM manager is also responsible for maintaining a correct link between the vTPM and its associated user VP, although this responsibility may be shared with the Control Domain or other trusted base components, and for passing correct measurements of the vTPM to the hardware TPM. Measurement responsiblity may be shared with the Domain Builder or other trusted base components.

The hardware TPM is used to ensure that the vTPM manager's state is itself up-to-date, and that the vTPM manager's secrets are preserved, as well as to provide support for vTPM data secrecy.

5 Provisioning a vTPM Manager

We rely on the vTPM Manager to correctly create and certify vTPMs. This is sometimes referred to as *vTPM Factory* functionality.[3] In order for a vTPM certification to be remotely verifiable, the vTPM Manager must have appropriate certificates itself.

[2] An adversary can cause a DoS on a vTPM by overwriting its data and making its secrets inaccessible, or even in some cases substituting new secrets, but cannot cause existing and certified secrets to be accessible to a "bad" system.

[3] A separate vTPM Factory component can be created to perform only these functions, potentially on a separate computer from the one the vTPM will be running on. We will not cover this scenario in this document.

5.1 AIC and other TPM certificates

The TPM must have an Endorsement Key (EK) and associated Endorsement Credential (EC). In addition, an Attestation Identity Key (AIK) must have previously been created and an Attestation Identity Credential (AIC) produced using the EC and signed by an appropriate certificate authority. The AIC will be provided to remote parties along with the vTPM manager and vTPM certificates.

Whatever party provisions the vTPM Manager must have access to the AIK.

5.2 vTPM Manager Key and Certificate

In order to be able to produce remotely verifiable certificates for vTPMs, the vTPM Manager must have its own key. We'll call this the *VFK*, short for vTPM Factory Key. The key should be a high-bit (1024 or more), non-migratable TPM signing key, with PCR constraints: it should only be accessible when the machine's PCR values correspond to a "good" vTPM Manager and a "good" trusted base.

(We guarantee that the VFK is only accessible to the vTPM Manager by a two-part inference: If the system has acceptable PCR constraints, then a good vTPM Manager and trusted base must be in place; and if a good vTPM Manager and trusted base are in place, then only the vTPM Manager itself has access to the TPM and therefore the VFK.)

Once the VFK has been created, it should be certified using the AIK and the TPM_CertifyKey command. This command produces an AIK-signed certificate testifying to the fact that VFK is a TPM key with particular attributes (e.g., non-migratable) and particular PCR constraints. We call this certificate the vTPM Factory Credential, or VFC.

When presenting vTPM certificates (endorsement credentials) to a remote party, the platform should include the VFC and AIC as well, allowing the remote party to trace trust back to a trusted certificate authority.

6 The vTPM Manager Table and Other Data

The core of the vTPM Manager's state is the vTPM Manager Table. This table is used to track the state of all vTPMs on the system; if the table becomes out of sync, then problems with the vTPMs may result.

The vTPM Manager Table should include the following information for each vTPM:

- The vTPM's unique identifier
- The platform ID associated with this vTPM; this should be a lifetime ID, not an ID for a single boot of the system

- A pointer to the VM image for the vTPM software, or other indication as to which software should run for this vTPM [4]
- A pointer to the location of the vTPM's data, if it is not easily derived from the ID
- A hash of the most recent vTPM data
- A TPM sealed key blob, containing the key that can be use to unlock the vTPM's data.
- Optionally, additional key blobs required to unseal the previous sealed blob, if more than a single vTPM Manager key is used
- Optionally, a set of PCR bindings (in the form of a PCR composite and optional locality constraint) that will be used to seal the vTPM's storage keys
- A flag indicating whether or not the vTPM is currently running

The vTPM Manager also preserves some data for its own use:

- A non-migratable TPM storage key blob, constrained to PCR values corresponding to a "good" vTPM Manager and trusted base, used to protect the vTPM Manager data[5]
- Optionally, a single storage key blob used for sealing all vTPM data storage keys. This key should be non-migratable, PCR locked to a "good" vTPM Manager and trusted base, and have the SRK as its parent.

7 vTPM Creation and Certification

We have three goals in the initial vTPM creation (which can also be called provisioning) and certification process:

1. A remote party should be able to verify that the certificate was produced by an "acceptable" system
2. An "acceptable" system should guarantee that a vTPM was freshly and correctly created
3. A correct creation results in the vTPM having booted correctly according to the requirements for long-term trust. [6]

[4] On systems where there is only expected to be one vTPM software image for the system lifetime, this can be skipped

[5] A hash of this key blob should be placed in TPM non-volatile storage by the platform owner for verification on boot.

[6] The trust argument works equally well if the post-creation state is equivalent to a safe shutdown, guaranteeing that it will boot correctly next time. If the easiest implementation path is to create a fresh vTPM as a new, static data blob instead of as a running VM, let us know and we can discuss the variant trust argument and its implications for, for example, how Endorsement Keys are created.

7.1 Creating the new vTPM

New vTPMs are created by vTPM Manager at the request of the Control Domain or other component responsible for launching virtual platforms. If a platform being launched does not already have a vTPM, as recorded in the vTPM Manager's table, but requires one, the vTPM Manager is responsible for creating one. [7]

For each new vTPM, the vTPM establishes the following information:

- A unique identifier for the vTPM, that will apply for its lifetime
- An association between the vTPM and its virtual platform, usually in the form of a long-term platform ID
- The vTPM software or VM image that will be used for this vTPM
- The initial settings for the vTPM; for a basic TPM emulator, this would include, for example, whether it begins activated or in FIPS mode.
- The correct PCR composite value for this vTPM, which will include the expected vTPM Manager and trusted base measurements as well as a measurement of the vTPM software itself in a resettable PCR. We will refer to this PCR composite as P.

We assume that a standard set of default settings for a vTPM is known to the vTPM Manager, or non-default settings can be used by request from the Control Domain.

Once the initial settings are established, the vTPM Manager should create initial vTPM state data and populate a vTPM Manager table entry for the vTPM. This state data should correspond to the vTPM settings, and should not contain anything that would not be expected from a TPM straight from the factory. (For example, a brand-new vTPM should not be created with an owner already in place.) A fresh symmetric encryption key should be created, used to encrypt the initial state data and sealed to P using the hardware TPM.

The vTPM can then be booted according to the normal vTPM Load procedure described in section 8.

7.2 Creating the vTPM EK

A vTPM's EK can be created in two ways: either the vTPM Manager can generate it as part of the TPM's initial state, just as a TPM Manufacturer can, or the vTPM can create it using the createEK command. We recommend the second approach, unless a specialized vTPM Factory is used, so as to minimize the complexity of the vTPM Manager and reduce the chance of a key being compromised during creation.

[7] In theory, a virtual platform may have multiple vTPMs. Some reasonable mechanism for determining when additional vTPMs should be created should be implemented in this case.

If the vTPM's own EK creation capability is used, then either the vTPM Manager or the user of the vTPM must have the vTPM execute the TPM_CreateEndorsementKey command. The resulting public key must then be transmitted by the vTPM to the vTPM Manager. The vTPM Manager must be certain that this message is from a specific vTPM, which successfully unsealed its data and therefore has the expected vTPM measurements, and that the message contains what the vTPM believes to be its own EK. We call the resulting key the virtual Endorsement Key, or vEK, for the vTPM.

7.3 Certifying the vTPM

The vTPM Manager creates a virtual Endorsement Credential (vEC) for the vEK using the same format that the TCG recommends for TPM manufacturers certifying hardware TPM Endorsement Keys, or any other format desired by the enterprise. This Credential is the vTPM Manager's claim that the vEK belongs to a genuine vTPM.

In addition, the vTPM Manager includes in the vEC the PCR composite P. This allows recipients of the vEC to determine whether or not P describes software they trust to behave as a vTPM.

The vEC should be presented to remote parties along with the VFC and AIC, in any circumstance in which a normal TPM might present an EC. These certificates can be provided to the vTPM directly, in the same fashion that some hardware TPMs provide a getCertificate function, or can be placed in some alternate location where the user VM can access them.

8 vTPM Load Steps

When a vTPM is launched, it must load its state from storage. We assume here that the Control Domain is responsible for sending the signal that a vTPM is to be launched, along with the platform ID it is to be associated with, and that the Domain Builder is responsible for measuring and launching the vTPM's VM image. Which components actually perform these roles does not actually matter for establishing trust, as long as the relevant components are included in the trusted base and therefore in the PCR values that control access to vTPM data.

1. The vTPM Manager receives a request from the Control Domain to launch the vTPM v for platform p.

2. The vTPM Manager consults its table and verifies that v is a legitimate vTPM id and corresponds to platform p.

3. The vTPM Manager retrieves the appropriate vTPM software image information for v from its table, and passes that information along with the ID v to the Domain Builder.

4. The Domain Builder measures and launches the vTPM software, and provides the vTPM with its ID as part of its launch information. The DB provides the vTPM Manager with the measurement m.

5. The newly launched vTPM requests the location of its data and its data storage key from the vTPM Manager. The vTPM Manager should confirm, using whatever mechanisms are available to it before the vTPM has access to its keys, that the VM asking for vTPM data is the vTPM in question; for example, it might compare the vTPM id to the VM type assigned by a Flask-style policy.

6. The vTPM Manager resets a hardware TPM PCR that it has access to (if not already reset) and extends the PCR with the measurement m. It then unseals the vTPM data storage key blob associated with v, producing an unsealed key k and guaranteeing that the desired PCR values are present.

7. The vTPM Manager provides k, the stored hash of the vTPM's data h, and, if necessary, the data storage location to the vTPM v, and updates the vTPM's status flag to "running".

8. The vTPM decrypts its data using k, verifying that the data integrity is intact using h, and makes itself available to its virtual platform.

We assume that some mechanism exists on the platform to inform the vTPM Manager when a vTPM is shut down, even unexpectedly, so it can update its status flag. However, the correct failure mode is for vTPMs of uncertain state to continue to be registered as "running"; a temporary DoS requiring a system reboot is safer than a situation in which two copies of a vTPM might be running simultaneously.

8.1 Verifying the integrity of vTPM data

vTPM data integrity can be ensured in several ways; any method that allows integrity verification given our adversary assumptions is acceptable. The simplest, described in this document, is to have the vTPM, upon saving its state (see section 9), take a hash of the data blob and provide it to the vTPM Manager. The vTPM Manager then provides this hash to the vTPM on launch, along with the decryption key and data location; the vTPM verifies the hash before attempting to decrypt the data.

8.2 Posssible Variations

- The Control Domain's initial request may not include the universal vTPM id; any request which uniquely identifies the vTPM in the table is acceptable. For example, in single-vTPM platforms, requests for "the vTPM for platform p" are reasoanble. In this case, the vTPM Manager identifies the universal id v in step 2.

- If an alternate mechanism vesides the vTPM id v exists to identify a vTPM uniquely to the vTPM Manager (such as a MAC type), the vTPM ID does not need to be passed to the Domain Builder and vTPM, and can potentially be used only as a table line identifier.
- The vTPM Manager may choose to always reset the vTPM measurement PCR immediately after unsealing a data key, in order to prevent accidental verification against that PCR, rather than or in addition to immediately before extending it with the new vTPM measurement.

9 vTPM SaveState Steps

Whenever the vTPM's non-volatile state changes, it must preserve a copy of its data. Non-volatile state includes any state that is preserved over a reboot, such as non-volatile flags, TPM ownership, TPM-resident keys (both inherent, such as the root keys, and owner-locked loaded keys), changes to the monotonic counter, and non-volatile storage. Note that loading keys and changing PCR values do *not* require that state be saved.[8]

1. vTPM generates a new symmetric key k for bulk encryption of its data.
2. vTPM encrypts its data, including all secrets and TPM state information such as non-volatile storage contents and flags, using k. It hashes the sealed data to create an integrity report h.
3. The vTPM stores the encrypted data on disk in location l
4. The vTPM transmits k, h and l, along with its vTPM identity id, to the vTPM Manager.
5. The vTPM Manager seals k using a hwTPM storage key. The blob k must be bound to the PCR values included in the original vEC for id.
6. The vTPM Manager updates its reference table entry for id with the new sealed k blob, h, and l.
7. The vTPM Manager saves its own new state.

This model of state preservation, in which any significant change is preserved when it happens, means that vTPMs are not obliged to perform a SaveState operation as part of shutdown. [9] Implementers may choose whether or not to do so.

[8] Volatile state is normally reset on platform boot in a hardware TPM. We assume here that although a virtual platform may be suspended or migrated, we do not wish to allow old vTPM PCR values to be preserved during this process. Other analyses and use cases may disagree. Also, some unusual variants of loading keys (those which only the owner can evict, for example, or which are deliberately designed to be maintained in the TPM) do require a saving of state; in general, save state whenever the changes should be maintained over a reboot.

[9] This is deliberate: crashed vTPMs should still boot properly later.

9.1 Atomicity Warning

Although these operations are not atomic, serious problems can be introduced if the process is interrupted partway through!

- A failure in steps 1-3 cause a state change to not be preserved.
- A failure in steps 4-7 would at best result in a key-data mismatch when the vTPM is loaded, resulting in a denial of service. In the worst case, if l is not constant, a failure to preserve an update to l, h, and k may result in a rollback to the vTPM's previous state.

9.2 Possible Variations

- Although it is necessary for the vTPM Manager to be able to tell future incarnations of the vTPM where the data is stored, the storage location l may be constant over time, in which case it is of course not necessary to retransmit the location on every SaveState operation.
- We assume here, for simplicity, that the vTPM Manager tracks the correct PCR bindings for each vTPM over the long term. The actual tracking of these values can be done by either the vTPM or the vTPM Manager, as long as the values are always consistent with the vEC. If the vTPM Manager is tracking these values, they should be stored as part of the vTPM Manager table.
- Although it is critical that the vTPM data stored on disk have some integrity protection, this can take forms other than the current integrity-verifying hash. Any integrity protection approach is acceptable, as long as an adversary with free access to the disk cannot cause a later vTPM boot to access certified vTPM keys without the correct, most recent vTPM state.
- The ordering of steps 3 and 4 can be reversed freely, or the steps can be performed in parallel.

10 vTPM Manager Load Steps

When the platform launches, the vTPM Manager must retrieve its most recent saved state.

The vTPM Manager's data is, like vTPM data, stored on disk encrypted with a symmetric key that is itself sealed using a TPM storage key. The blob for that TPM storage key can be read from the TPM's owner-writable non-volatile storage[10] and loaded by the vTPM Manager. Use of this storage key should require a set of PCR constraints describing a "good" trusted computing

[10] Alternately, if space is tight, a hash of the blob can be put in NV storage instead, and the blob itself stored in a known location on disk.

base and vTPM Manager, ensuring that vTPM Manager data is not decryptable on a bad boot. The vTPM Manager should also retrieve from the TPM's non-volatile storage hashes of its encrypted data and sealed symmetric key, and verify that the hashes match the data on disk.

Once the storage key is loaded, the symmetric key can be unsealed and used to decrypt the vTPM Manager data, establishing the current state for the vTPM Manager table. All "running vTPM" flags in the vTPM Manager table, and any other volatile data, should then be cleared. The vTPM Manager is now ready to handle requests.

11 vTPM Manager SaveState Steps

The vTPM Manager should perform the SaveState operation whenever a change is made to its internal data that should be maintained over a boot cycle. This includes all updates to vTPM data keys and storage locations, as well as the creation or deletion of vTPMs. It does not include changes to the "running vTPM" flag, as that flag is volatile and reset on every boot.

1. The vTPM Manager first creates a fresh symmetric key, and uses it to encrypt the vTPM Manager Table contents, vTPM Manager keys, and any other internal data.
2. The vTPM Manager loads its TPM storage key found in the TPM if necessary, and seals the symmetric key with it. [11]
3. The vTPM Manager updates its stored data location with the encrypted data and sealed symmetric key blob.
4. It then takes a hash of the encrypted data and the sealed symmetric key blob, and puts them into a section of the TPM's non-volatile storage that is writable only when a "good" vTPM Manager and trusted base are running. [12]

11.1 Atomicity Warning

As with the vTPM data, the SaveState operation is not atomic, but failures in the middle of the operation can cause problems. Failure in the first two steps causes the state update to not be preserved; failure in the last two steps can cause the data to not be accessible.

[11] Because the storage key has PCR constraints, no additional constraints on the sealed data blob are necessary.

[12] This should be the same set of PCR constraints used for the data storage key and the VFK.

12 Migration and Other Hypotheticals

This architecture should easily support vTPM migration as part of platform migration, although a vTPM (unlike a normal VM) may need to be aware of its own migration in order to properly prepare its secret data for transfer. A vTPM can store its data using a migratable TPM key, rather than the normal non-migratable TPM key, although the storage key must still preserve PCR bindings. The vTPM Manager can then run a vTPM migration protocol, providing the vTPM data and the vTPM storage key.

Although we have previously described a vTPM migration protocol, we have not yet integrated it with this vTPM architecture or the hardware TPM; implementers may, however, wish to be aware that such plans are under development for the future.