vTPM Modeling Notes

Perry Alexander and Brigid Halling
Information and Telecommunication Technology Center
The University of Kansas
{palexand,bhalling}@ku.edu

Allen Goldberg, Eric Smith, Alessandro Coglio
The Kestrel Institute
{addresses}@kestrel.edu

June 18, 2013

Contents

List of Figures

List of Tables

Abstract

This document is designed to document our work understanding the interaction among elements of the vTPM infrastructure and interaction between the vTPM and its operational environment.

1 Introduction

2 vTPM Manager Data

Data objects used by the vTPM subsystem are listed in figure 1 and figure 2. Figure 1 lists data that is stored by the vTPM Manager and includes PCR contents associated with vTPM and controller measurements, long- and short-term IDs, the sealed vTPM Data key and vTPM Data hash.

Field	*Description*	*Initialized*	*Updated*
ID_{vtpm}	The persistent name of the vTPM	provisioning	never
ID_{dom}	Domain ID of vTPM VM	build	build
$\#D$	vTPM data hash value	provisioning	vTPM sleep
PCR_6	Hash of the vTPM + Long Term Name	provisioning	boot
PCR_5	Hash of the platform controller	provisioning	boot
$seal(K_D, \{PCR_5, PCR_6\})$	Sealed symmetric key encrypting vTPM data	provisioning	vTPM sleep

Figure 1: vTPM Information Table contents

Figure 2 lists data that is involved in vTPM related communications and includes the encrypted vTPM data, the vTPM image, and hashes of the controller and booted vTPM. Note that the hash values are called out separately here as they may differ from what is stored in the vTPM Manager.

Data	*Description*	*Location*	*Initialized*	*Modified*
$\{D\}_{K_D}$	Encrypted vTPM data	Host Storage	provisioning	vTPM sleep
I_{vtpm}	*vTPM Image*	Host Storage	provisioning	never
$\#(vTPM + ID_{vtpm})$	Hash of vTPM named ID_{vtpm}	vTPM Manager	build	build
$\#Controller$	Hash of Controller and Schema	vTPM Manager	build	build

Figure 2: Other data used during vTPM activities

3 Provisioning Sequence

I'm not at all sure I have the provisioning sequence right. The vTPM can be used to generate an EK while the TPM_TakeOwnership command can be used to generate an SRK after startup. What else is initialized? Monotonic counter values and other stuff not specified here.

1. Platform booted through Controller – PCR_5 is now known by the vTPM Manager
2. A new Long Term Name is generated for the new vTPM
3. The new vTPM is built using the Domain Builder – PCR_6 is now known by the vTPM Manager

4. vTPM data is initialized – including fresh EK value.
5. vTPM is put to sleep using the standard sleep protocol

Pre-conditions for Provisioning

Precondition 1 (MLE Measurements) *hTPM logical PCR_{0-4} contain hashes of MLE components.*

Precondition 2 *hTPM logical PCR_5 contains a hash of the controller/schema pair uniquely identifying the vTPM's virtual platform.*[1]

Post-conditions for Provisioning

vTPM Information Table

Conditions on the vTPM information table residing in the vTPM Manager:[2]

Postcondition 1 (Controller Measurement) *The vTPM's controller measurement is available in the vTPM information table referenced as PCR_5.*

$$PCR_5(ID_{vtpm}) = \#Controller$$

Postcondition 2 (vTPM Measurement) *The vTPM's measurement is unique and available in the vTPM information table, referenced as PCR_6.*

$$PCR_6(ID_{vtpm}) = \#(vTPM + ID_{vtpm})$$

Postcondition 3 (vTPM Data Existence) *The vTPM's initial data exists and is available through its table entry.*

$$\forall i : ID_{vtpm} \cdot \exists d : D \cdot \{d\}_{K_d(ID_{vtpm})}$$

Postcondition 4 (vTPM Data Seal) *The vTPM's data will be sealed to hTPM logical PCR_{0-6}*

Postcondition 5 (vTPM Data Hash) *The vTPM's data hash will be stored in the vTPM information table.*

Postcondition 6 (Endorsement Key) *The vTPM's Endorsement Key (EK) is initialized and a certificate binding it to its vTPM is available.*

$$\exists k : KEY \cdot EK(ID_{vtpm}) = k$$

$$\exists [\![EK(D)]\!]_{CA} : cert \cdot TRUE$$

Postcondition 7 (Unique vTPM Name) *The ID_{vtpm} will be unique.*

$$\forall i, j : ID_{vtpm} \cdot t(i) = t(j) \Longleftrightarrow i = j$$

[1] The controller and schema may be hashed separately with the schema providing the unique identifier for the platform.

[2] Note that we use the PVS notation $X(ID_{vtpm}$ to reference property X from the vTPM Manager's record for vTPM ID_{vtpm}.

3.1 Invariants Following Provisioning

3.2 Conjectures

Conjecture 1 (vTPM Data Seal) *A vTPM cannot access its data if its platform was not built from the correct parts.*

The state of a vTPM's data session key following provisioning or sleep is sealed to PCR_0 through PCR_6 that contain measurements of MLE and SVP components that boot before the vTPM:

$$seal(K_D, \{PCR_0 \dots PCR_6\})$$

Given the key is initially sealed during provisioning to $PCRs$ resulting from hashing correct components, K_D will not unseal unless: (i) the unsealing TPM is the same as the sealing TPM; and (ii) hashes in sealing $PCRs$ must match hashes of those in the TPM at provisioning. Assumption (i) and a certificate binding the sealing TPM to a system ensures that the K_D will only unseal on the hardware it was sealed on. Assumption (ii) and the assumptions that the hash function is injective and the MLE is correct ensures that K_D will only unseal if the correct Domain Builder, vTPM Manager, Controller and Schema, and vTPM were started during system boot.

The assumption that ID_{vtpm} values are unique ensures that K_D will not unseal unless the correct vTPM makes the unseal request. The vTPM Manager resets PCR_5 and PCR_6 based on the domain ID (ID_{dom}) of vTPM initiating communication.

4 Startup Sequence

Establishing an operational vTPM involves getting the vTPM up and running, then installing its data. These are outlined in the following to subsections.

4.1 vTPM Running

Figure 3 describes the sequence of events necessary for a vTPM to start. The result of this sequence is a vTPM running and the vTPM Manager table reflecting vTPM information. The vTPM will not have its data yet. Following is a textual description of the process:

1. Controller requests a build by sending a manifest identifying the vTPM image and its Long Term Name to the Domain Builder
2. Domain Builder hashes the vTPM image and Long Term Name
3. Domain Builder starts the vTPM VM resulting in a Domain ID
4. Domain Builder sends the Long Term Name, Hash, and Domain ID to the vTPM Manager
5. vTPM Manager updates data associated with the Long Term Name in the vTPM information table

4.2 vTPM Data Initialization

Figure 4 describes the sequence of events necessary for a vTPM to load its data. The result of this sequence is a vTPM ready to provide TPM services to its associated virtual platform. Following is a textual description of the process:

1. vTPM requests its data key from the vTPM Manager
2. vTPM Manager uses the vTPM Domain ID to find table entry

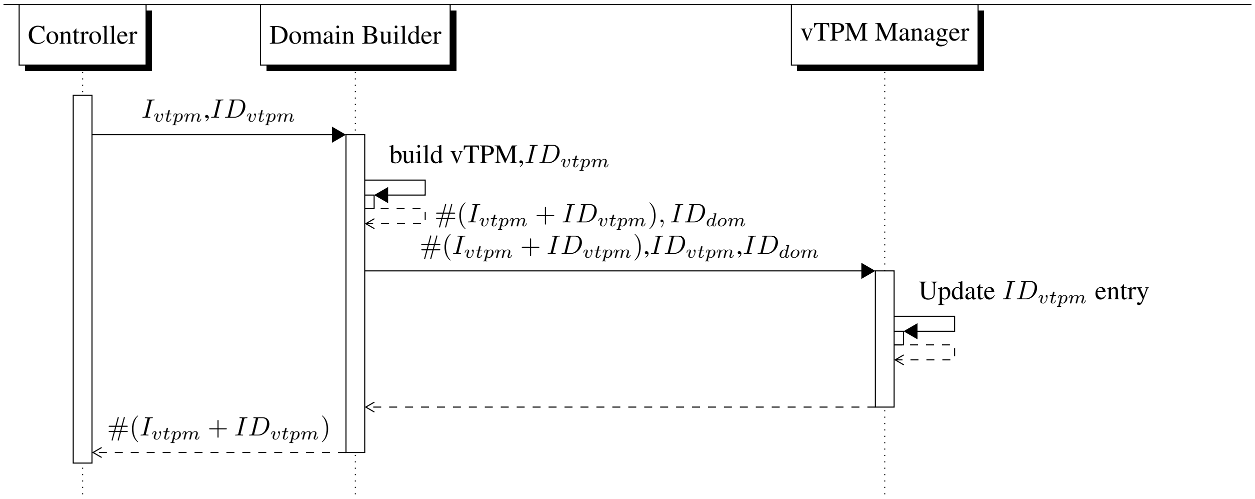
Controller
Domain Builder
vTPM Manager
I_{vtpm},ID_{vtpm}
build vTPM,ID_{vtpm}
$\#(I_{vtpm} + ID_{vtpm}), ID_{dom}$
$\#(I_{vtpm} + ID_{vtpm})$,ID_{vtpm},ID_{dom}
Update ID_{vtpm} entry
$\#(I_{vtpm} + ID_{vtpm})$

Figure 3: Command sequence to establish a running vTPM without its data.

3. vTPM Manager resets PCR_5 and PCR_6 with table entry values[3]
4. vTPM Manager attempts to unseal K_D
   (a) On failure abort the request response
   (b) On success return K_D and $\#D$ to vTPM
5. vTPM requests its encrypted data from Host Storage
6. vTPM decrypts its data with K_D and checks hash against $\#D$
7. vTPM installs data and begins providing services.

5 Sleep Sequence

Figure 5 describes the sequence of events necessary for putting a vTPM to sleep. The result of this sequence is the vTPM's data saved to Host Storage encrypted with a fresh session key. The vTPM Manager's table will be updated to reflect the new key and data hash. Following is a textual description of the process:

1. vTPM generates a fresh session key, K_D
2. vTPM hashes its data, $\#D$ and encrypts with K_D
3. vTPM stores $\{D\}_{K_D}$ in Host Storage
4. vTPM sends sleep request to vTPM manager with K_D and $\#D$
5. vTPM Manager uses vTPM Domain ID to find table entry
6. vTPM Manager resets and loads PCR_5 and PCR_6 from table entry
7. vTPM Manager seals K_D to TPM state
8. vTPM Manager updates its table with $seal(K_D, \{PCR_5, PCR_6\})$ and $\#K_D$
9. vTPM Manager clears Domain ID[4]

[3] This happens whenever the vTPM Manager communicates with a vTPM.

[4] I'm not certain of this, but it seems reasonable

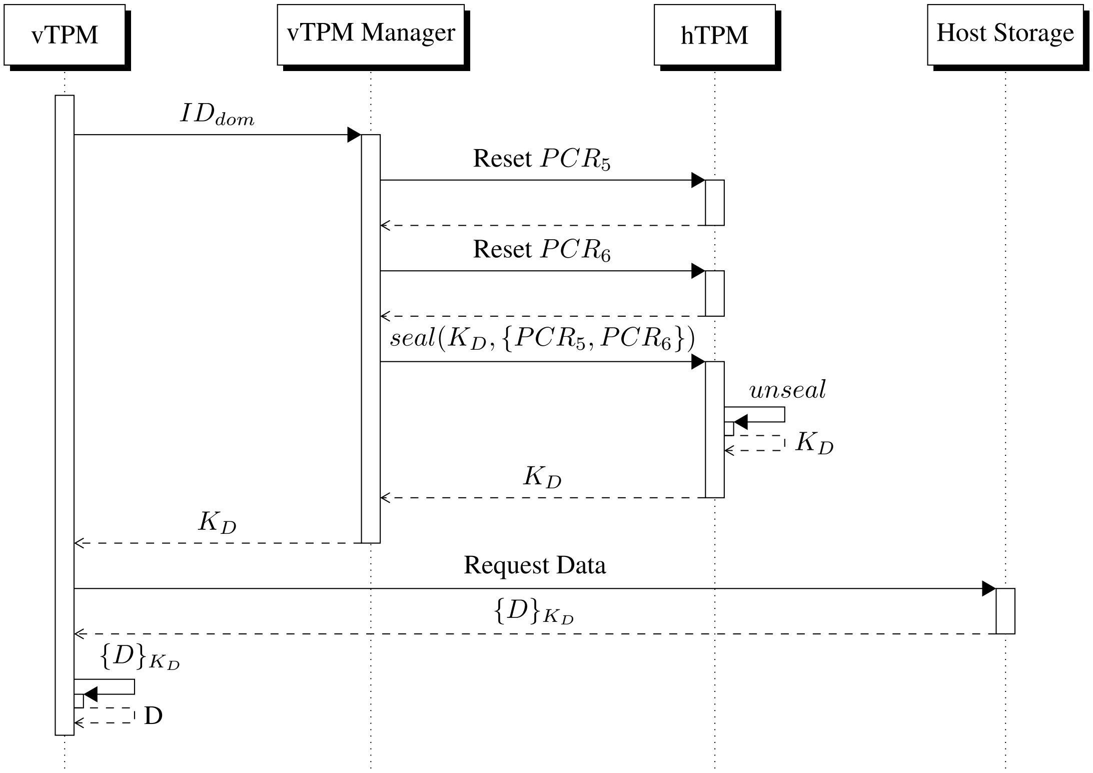

Figure 4: Command sequence to load vTPM data.

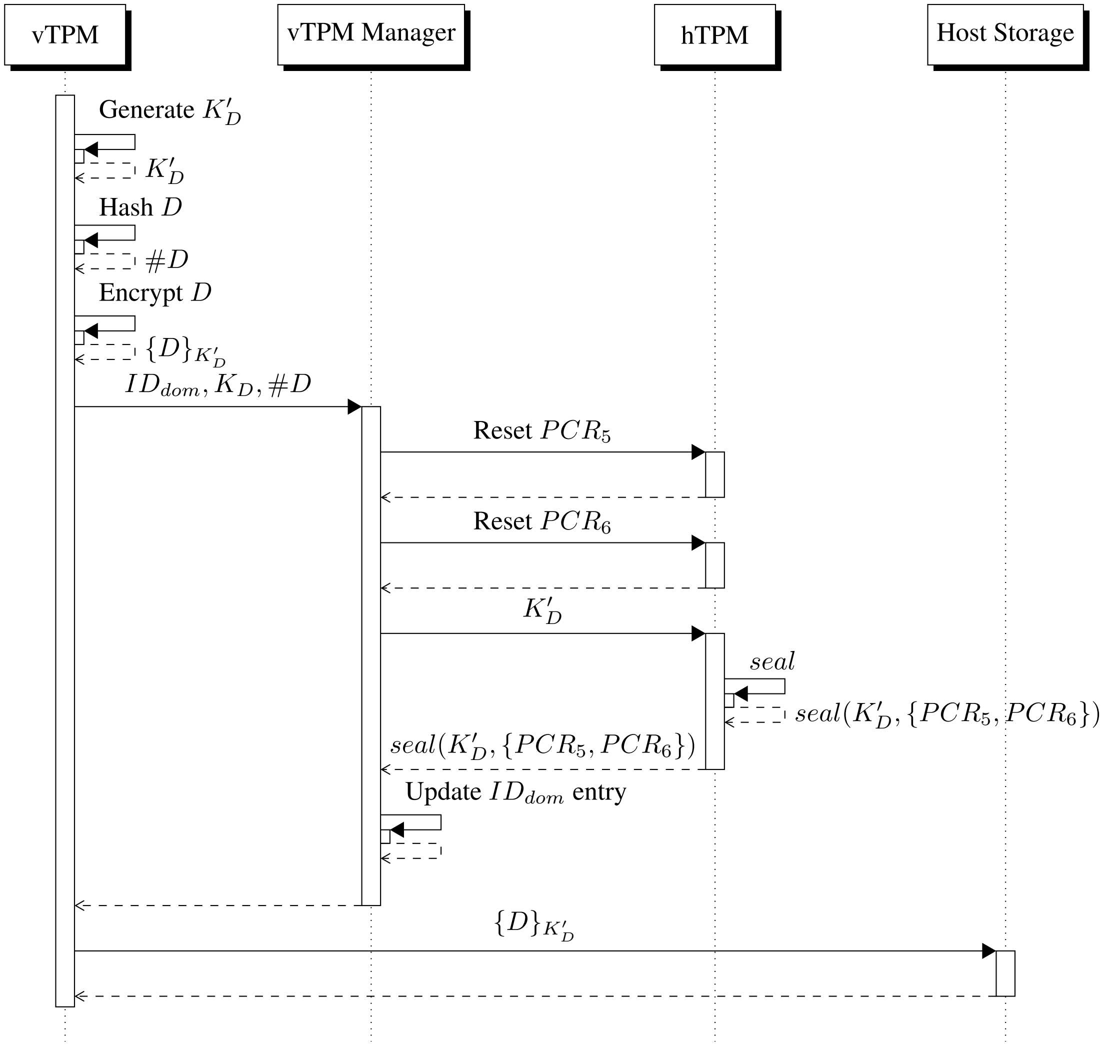

Figure 5: Command sequence to put a vTPM to sleep.