TPM Specification Design

Perry Alexander Brigid Halling

June 27, 2012

Contents

List of Figures

```
tpmAbsState : TYPE = [#
                     srk : (asymKey?),
                     ek : (asymKey?),
                     keys : KEYSET,
                     pcrs : PCRS,
                     locality : LOCALITY
                     #];
```

Figure 1: Abstract TPM state record data structure.

List of Tables

Abstract

The abstract goes here...

1 Introduction

2 Modeling Approach

2.1 TPM State Monad

Figure 1 is the PVS record structure used to represent the internal state of the TPM.

2.2 TPM Command Definitions

Figure 2 is the PVS data type used to represent the abstract syntax of the TPM command set.

Table 1 maps TPM concrete commands to their abstract PVS representations.

Table 2 maps external commands that interact with the TPM to their PVS representations.

```
tpmInput : DATATYPE
BEGIN
  decryptKeyCom(d:(encrypt?)) : decryptKeyCom?
  encryptCom(b:BLOB) : encryptCom? % unimplemented
  extendCom(h:HV,n:PCRINDEX) : extendCom?
  installKeyCom(k:(wrapKey?)): installKeyCom?
  noopCom : noopCom?
  offCom : offCom?
  powerCom : powerCom?
  quoteCom(n:BLOB,pm:PCRMASK) : quoteCom? % partially implemented
  revokeKeyCom(k:(wrapKey?)): revokeKeyCom?
  senterCom : senterCom? % implemented all actions as one senter
  sinitCom : sinitCom? % partially implemented
  unsealCom(d:(seal?),k:(asymKey?)) : unsealCom?
END tpmInput;
```

Figure 2: TPM command data type.

TPM Command	*Abstract PVS Command*	*Concrete PVS Command*
TPM_CreateWrapKey		
TPM_LoadKey2	`instalKeyCom(k)`	
TPM_Seal		
TPM_Unseal	`unsealCom(d,k)`	
TPM_Extend	`extendCom(h,n)`	
TPM_Quote	`quoteCom(b,pm)`	
TPM_MakeIdentity		
TPM_ActivateIdentity		

Table 1: TPM command mapping to PVS command representation.

External Command	*Abstract PVS Command*	*Concrete PVS Command*
senter	`senterCom`	
sinit	`sinitCom`	
Power On	`powerCom`	
Power Off	`offCom`	

Table 2: System commands interacting with TPM.

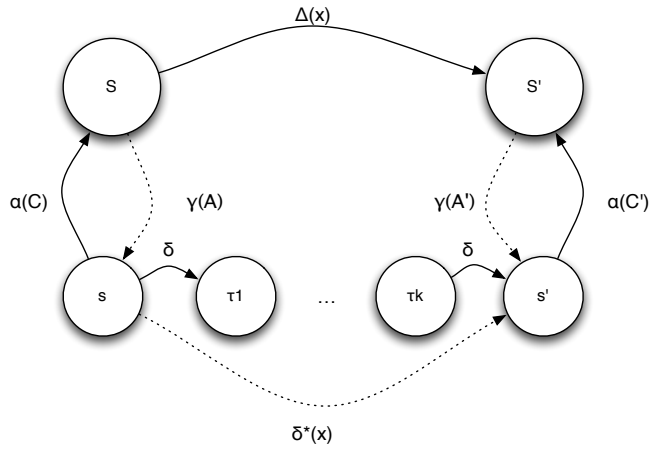

Figure 3: Weak bisimulation relation between an abstract transition system $A = (S, \Sigma, \Delta)$ and a concrete transition system $C = (s, \sigma, \delta)$.

3 Verification Approach

The approach taken for verification is establishing a *weak bisimulation* [Sangiorgi, 2012] relation between an abstract requirements model and a model derived from the TPM specification as shown in figure 3.

We say that $A = (S, \Sigma, \Delta)$ is an *abstract model* where S is a set of abstract states, Σ is a set of actions on states and input, and $\Delta : S \times \Sigma \rightarrow \Sigma$ is a transitions on state and action. Similarly, we say that $C = (s, \sigma, \delta)$ is a *concrete state* where s is a set of concrete states, σ is a set of actions on states and input, and $\delta : s \times \sigma \rightarrow \sigma$ is a transition function.

We relate the abstract and concrete models through an *abstraction function*, $\alpha : s \rightarrow S$, and *concretization function*, $\gamma : S \rightarrow 2^s$. The abstraction and concretization functions must form a Galois Connection such that:

$$s \in \gamma(\alpha(s))$$

Specifically, when making the result of an abstraction concrete, the original state must be in the resulting set. Note that the concretization function may result in multiple states due to the necessity of specifying unknown detail.

We say that A and C are weakly bisimilar ($A \sim C$) if when $\alpha(s) = S$ then $\alpha(\delta^*(s)) = \Delta(S)$ for all inputs to s.

Definition 1

$$A \sim C \equiv \forall s_0 : s \cdot \exists S_0 : S \cdot \alpha(s_0) = S_0 \Rightarrow \alpha(\delta^*(s_0)) = \Delta(S_0)$$

In the formal TPM model `tpmAbsState` defines S while `tpmConcState` defines s.

Glossary

This glossary is intended to document some common acronyms as well as define some common terms. It is currently a bit haphazard and a number of elements are missing.

A Trusted Platform Definitions

Trusted Platform Module (TPM) – Hardware Trusted Platform Module as defined by TCG.

Process Configuration Register (PCR) – Registers defined in the TPM. In the TPM there are at least 16 and they are 20 bytes wide.

Bound to PCR – Encryption including PCR values that must match TPM PCR values before decryption is performed.

Sealed to State – Operation performed by the TPM where data is encrypted and bound to PCRs or a PCR composite that must be checked before unsealing. Only data is sealed.

Wrapped Key – An asymmetric key with its private key encrypted and its public key visible. Only keys are wrapped.

Virtual TPM (vTPM) – A virtual Trusted Platform Module

Certified Key (CK) - Asymmetric key with private key signed by the AIK private key.

Endorsement Key (EK) - Asymmetric key who's private key is in the TPM hardware. Private key is used to sign data from TPM while public key is used to encrypt sensitive data sent to the TPM. The EK is set by the TPM factory and may not be reset.

Attestation Identity Key (AIK) - Asymmetric key whose private key is used for only two purposes: (i) sign (or attest to) TPM internal state; (ii) sign (or certify) other general purpose keys. AIKs are generated by a TPM, certified by a trusted third party, and used for signing in lieu of the EK.

Storage Root Key (SRK) - Root of secure storage hierarchy. Used to encrypt storage keys that exist outside the TPM. The SRK may be reset.

Attestation Identity Certificate (AIC) - Certificate provided by a trusted third party binding an AIK to a specific trusted platform.

Digest - 20-byte Value contained in a PCR.

PCR Composite - Single digest value generated from a collection of PCR values.

Quote - A value along with a set of PCR values or PCR composite signed by a TPM using an AIK.

Root of Trust for Measurement (RTM) - the "place to stand" for measurement. Effectively a hardware-based trusted launch point that will faithfully measure, start and pass control to its target without trusting other components.

Root of Trust for Storage (RTS) - the "place to stand" for measurement storage. Effectively a trusted hardware store that will store measurements with integrity without trusting other components.

Root of Trust for Reporting (RTR) - the "place to stand" for generating quotes and providing evidence with integrity and authenticity. Effectively a trusted hardware component that generates and signs quotes.

B Cryptography Notations

Hash Notation $\#data$ - The hash of $data$.

Certificate Notation $[[cert]]_{key}$ - Certificate, $cert$, signed by key.

Signed Data Notation $\{|data|\}_{key^{-1}}$ - $data$, signed by key.

Encrypted Data Notation $\{data\}_{key}$ - $data$ encrypted with key.

Sealed Data Notation $seal(data, \{pcrs\})$ - $data$ sealed to $pcrs$.

Wrapped Key Notation $wrap(k, \{pcrs\})$ - Equivalent to the key pair $(seal(k^{-1}, \{pcrs\}), k)$

C Intel Secure Boot

Trusted eXecution Technology (TXT) - Intel's trusted boot support.

Measured Launch Environment (MLE) - Run time environment providing a measured boot. Initialized and started by the SINIT command execution.

SENTER or GETSEC - Intel's trusted boot command. Provides synchronization, special bus cycles, and a special environment residing on the CPU (ACEA).

Secure INITialization Instructions (SINIT) - Code for performing secure initialization. Loaded by SENTER and validated by the ACM.

Authenticated Code Execution Area (ACEA) - CPU resident environment for executing code known as the Authenticated Code Module (ACM). Boot sequence involving SENTER and SINIT is as follows:

1. Load SINIT and MLE into memory
2. Invoke SENTER (GETSEC)
3. Establish special environment (ACEA)
4. Load SINIT into ACEA

5. Validate SINIT digital signature and store SINIT identity in TPM
6. SINIT measures MLE in memory and stores MLE identity in TPM
7. SINIT passes control to MLE

Authenticated Code Module (ACM) - Code running in the ACEA. May be used for validating platform configuration, measuring the measured launch environment, cleaning up after crashes.

References

D. Sangiorgi. *Introduction to Bisimulation and Coinduction.* Cambridge University Press, 2012.