TPM 2.0 TSS System API and Test Application

Will Arthur
Intel Corporation
7/7/15

About

The TPM 2.0 TSS system API library code implements the system layer API level of the TSS 2.0 specification. These functions can be used to access all TPM 2.0 functions as described in Part 3 of the TPM 2.0 specification. The usefulness of this code extends to all users of the TPM, even those not planning to use the upper layers of the TSS.

Between the system API library code and the TPM, there is a TAB/RM (TPM Access Broker/Resource Manager) daemon that handles all multi-process coordination and manages the TPM's internal resources transparently to applications.

The TAB/RM daemon communicates through linked in driver code to communicate with the TPM 2.0 simulator.

Additionally, the test application, tpmclient, tests many of the commands against the TPM 2.0 simulator. The tpmclient application can be altered and used as a sandbox to test and develop any TPM 2.0 command sequences, and provides an excellent development and learning vehicle.

Instructions

To use the TPM 2.0 library, simulator client, and test app:

1. Build and test TPM 2.0 simulator:
    a. Get the TPM 2.0 simulator, 1.24 version, from the TCG web site, trustedcomputinggroup.org, and install it (this is only possible if you or your company is a TCG member):
        i. Go to www.trustedcomputinggroup.org
        ii. Click on the member login link (you will have to sign up if you aren't already)
        iii. Go to groups→ TPMWG→ Documents → Filter, and search for "TPM 2.0 VS". The latest one as of this writing is 1.24.
        iv. This will give you the source code, so you will have to follow the instructions for building it.
        v. In Visual Studio 2012, open TPMcmd\simulator.sln solution file and build it.
    b. Copy libeay32.dll (from OpenSSL) into TPMcmd\debug directory.
    c. Run TPMcmd\debug\simulator.exe

d. To test it the following python script can be used.

NOTE: you may have to cut and paste these commands into Python interpreter one by one. I'm not a python expert, and I couldn't get the script to just run:

```
import os
import sys
import socket
from socket import socket, AF_INET, SOCK_STREAM

platformSock = socket(AF_INET, SOCK_STREAM)
platformSock.connect(('localhost', 2322))
platformSock.send('\0\0\0\1')

tpmSock = socket(AF_INET, SOCK_STREAM)
tpmSock.connect(('localhost', 2321))
# Send TPM_SEND_COMMAND
tpmSock.send('\x00\x00\x00\x08')
# Send locality
tpmSock.send('\x03')
# Send # of bytes
tpmSock.send('\x00\x00\x00\x0c')
# Send tag
tpmSock.send('\x80\x01')
# Send command size
tpmSock.send('\x00\x00\x00\x0c')
# Send command code:  TPMStartup
tpmSock.send('\x00\x00\x01\x44')
# Send TPM SU
tpmSock.send('\x00\x00')
# Receive 4  bytes of 0's
reply=tpmSock.recv(18)
```

2. Build system API and test code:
   a. Windows:
      i. Create an environment variable, TSSTOOLS_PATH that points to your Visual Studio C nmake.exe, cl.exe, link.exe, and lib.exe utilities. NOTE: the path should start and end with double quotes so that any spaces in the path are interpreted properly.
      ii. In Visual Studio 2010, open the tss.sln solution file.
      iii. Build it. This will build the System API library (tpm.lib), the TAB/RM daemon (resourcemgr.exe) and the test application (tpmclient.exe). tpm.lib is linked into both the TAB/RM daemon and the test application. The test\tpmclient\debug or test\tpmclient\release directories are where the resourcemgr.exe and tpmclient.exe files are located after building, depending on the type of build.

iv. Start the TPM 2.0 simulator (this assumes you have a working version of this installed).
v. Start the TAB/RM daemon, resourcemgr\<debug | release>\resourcemgr.exe. There are command line parameters for selecting the TPM host and ports and the ports that will be used by applications to communicate with the daemon. The easiest way is to run everything on the same machine, in which case no command line parameters are needed. For help with the command line parameters, type "resourcemgr -?". For comparing to known good output, redirect the resourcemgr output to a file.
vi. Run the test\<debug | release>\tpmclient.exe application. It will test various TPM 2.0 library functions. Redirect the output to a file: tpmclient -host –dbg 3 out_file 2>&1. Type "tpmclient -?" to see help text that describes the debug message levels and other command line options.
vii. Compare the output file to the good sample output files, test\tpmclient\good\out.good and test\tpmclient\rm.out.good. The same number of commands should have run and pass/fail status should be the same for all the tests. There will be miscompares due to randomness in some forms of output data from the TPM, but these are not errors.

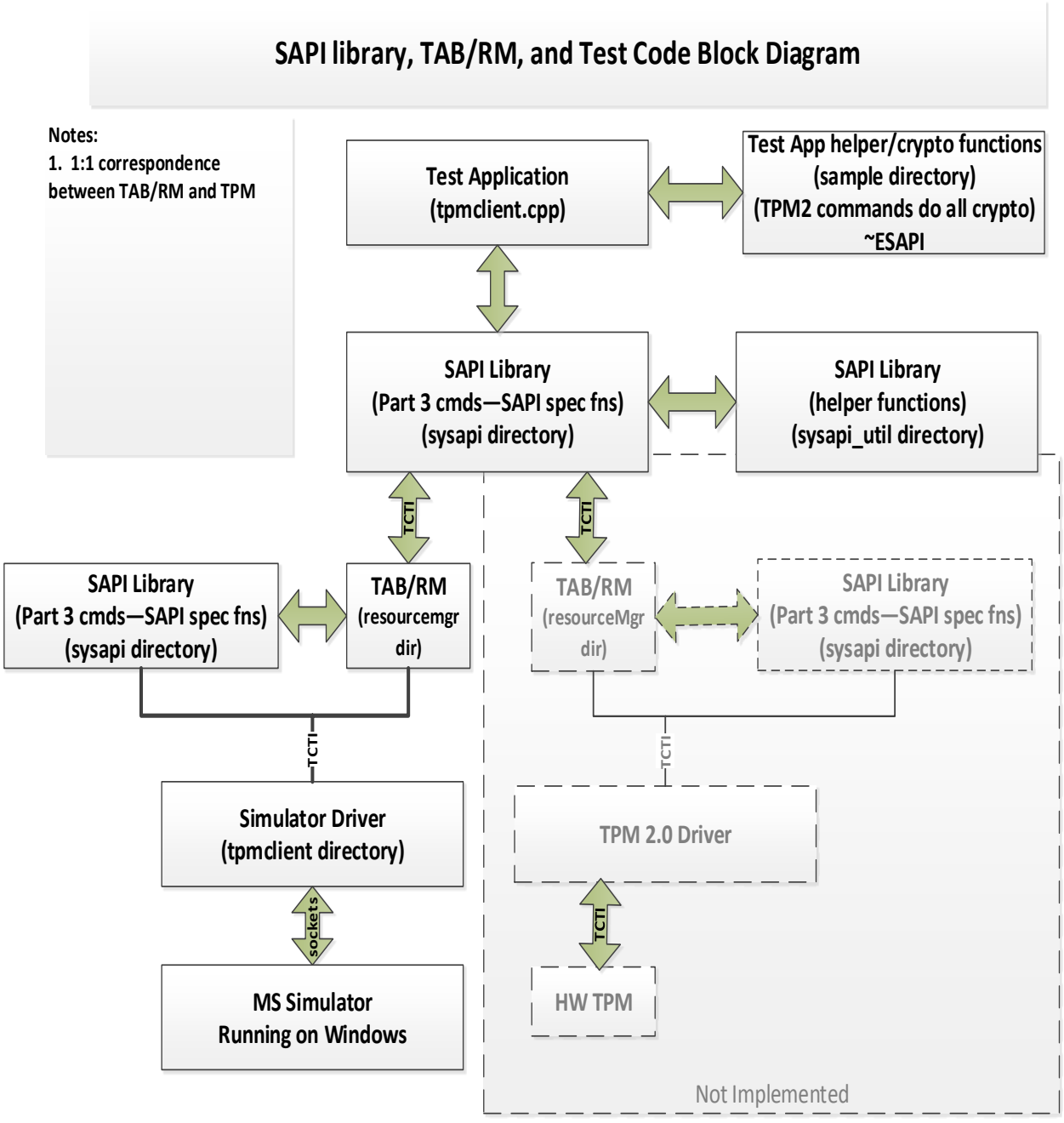
b. Linux:
i. Follow the build and install instructions in the INSTALL file.
ii. Follow steps vii and viii as detailed above for Windows with slight modifications for running under Linux.

Reporting Bugs

Intel requests that any bugs found in this code and bug fixes be reported to Intel so that all users of this code can benefit. Please report any issues to: Will Arthur, will.c.arthur@intel.com.

Architectural Block Diagram

This shows the blocks and interfaces in the SAPI library, TAB/RM, and test code.



Code Layout

Below is a diagram of the directory structure for the code under Windows. For Linux, libraries and executables are located differently: