TPM 2.0 TSS System Api and Test Application

Will Arthur
Intel Corporation

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.

Additionally, the test application, tpmclient, tests many of the commands against the TPM 2.0 simulator. A sockets interface and a built-in TPM 2.0 simulator interface driver are used in the application to communicate with 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:
    2. Get the TPM 2.0 simulator, 1.15 or 1.19 version, from the TCG web site, trustedcomputinggroup.org, and install it.
    3. Unzip the files into a root directory.
    4. Build and test the simulator:
        a. Follow directions in the simulator release notes to build the simulator.
        b. In Visual Studio 2012, open TPMcmd\simulator.sln solution file and build it.
        c. Copy libeay32.dll into TPMcmd\debug directory.
        d. Run TPMcmd\debug\simulator.exe
        e. 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)
```

5. Build system API and test code:
    a. Windows:
        i. Create an environment variable, TSSTOOLS_PATH that points to the base directory of your Visual Studio install.
        ii. In Visual Studio, open the tpm\test\tpmclient.sln solution file.
        iii. Build it. This will build both the System API library and the test application.
        iv. Start the TPM 2.0 simulator (this assumes you have a working version of this installed).
        v. Run the tpmclient.exe file. It will test various TPM 2.0 library functions. Redirect the output to a file: tpmclient -host <hostname> -port <port #> > out_file 2>&1. You can also use the "–dbg <debug level>" command line option to control the amount of debug messages. Type "tpmclient -?" to see help text that describes the debug message levels.
        vi. Compare output file to the good sample output file, test\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. Build library code from root direction of system API code: > make –f makefile.linux
        ii. Change to Test/tpmclient directory
        iii. Build test code: >make –f makefile.linux
        iv. Run the tpmclient executable and redirect the output to a file: >tpmclient –host <hostname> -port <port #> > out 2>out. You can also use the "–dbg <debug

level>” command line option to control the amount of debug messages. Type “tpmclient -?” to see help text that describes the debug message levels.

v. Compare output file to the good sample output file, test\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.

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.