EMULATING TRUSTED PLATFORM MODULE 2.0 ON RASPBERRY PI 2

Jen-Liang Cheng, Kuang-Chi Chen, Hong-Wei Zhang, Wei-Yu Chen and Dai-Ye Wu

Department of Medical Informatics, Tzu-Chi University, Hualien City, Taiwan

ABSTRACT

A computer hijacked by a malware may pretend that it is normal as usual and retrieve secrets from storage of itself and other victim computers. By adopting trusted computing technology a computer's former health status cannot be forged. Computers can thus detect the change of health status of a hijacked computer and prevent the leakage of the secrets. As Trusted Computing Group (TCG) proposed Trusted Platform Module (TPM) specification, IBM implemented software TPM (sTPM) and utilities for engineer who wants to learn the operating principle of TPM. Meanwhile, the blooming of tiny size, but powerful, computers, e.g. Raspberry Pi 2 (Rpi2), attract ones to develop some dedicated applications on the computers. In this article, we report the verified steps for installing new sTPM version on RPi2. After the installation, we also test the functionality and evaluate the performance of the sTPM with some major TPM Commands. The real behaviour of and the traffic between the host computer and the emulated TPM can thus be learned easily.

KEYWORDS

Trusted Computing, Trusted Platform Module, Software TPM, Raspberry Pi 2.

1. INTRODUCTION

A computer consistently behaves in expected ways is called a trustable computer. To assure that, the health status of the computer must be verifiable and unforgeable. This requires a hardware component that reports the health status encrypted with its private key inaccessible from the rest of the hardware component. Other computers can then verify the health status with the associated public key. The health status of the computer is recorded in several status registers as described. When a computer's power is on, the very beginning of the booting ROM, called root of trust, hashes the rest code of boot ROM and records the hash result in the first register. Then the booting block on the hard disk is hashed along with the previous result and recorded in the second register for health status. Then the code of operating system is hashed along with the result in the second register. This process goes on until all software sections that needs to be verified are hashed. The process is called 'chain of trust'. The health status can only be hashed and not writable, i.e. a malware is unable to forge health status. For example, if the operating system, for example, is contaminated, the health status related to the operating system must be different from the original one after the next booting process. The malware can neither forge the status report due to that it does not know the private key of the component. When a computer is hijacked, the change of its health status can thus be detected. Furthermore, secrets can be stored into the component under a certain health status and be accessed only when the computer is under the same health status.

Above example depicts a simple application of trusted computing. The concept about the hardware component mentioned is achieved through Trusted Computing (TC) technology. In reality, the core technology of TC is the cooperation of endorsement key, sealed storage, and remote attestation. To enhance security, TC technology is suggested to be implemented on a chip, named as Trusted Platform Module (TPM), whose specification is proposed by Trusted Computing Group (TCG). Currently the specification is TPM2.0 and is not backward compatible to its previous version, TPM1.2.

TPM 1.2, as shown in Figure 1, can be treated as a system on chip with interface of serial peripheral interface (SPI). Ideally, a TPM is shipped with a private key that is certified by its manufacturer. This key is not accessible by any software outside of the chip. Based on the unique private key, sensitive data can only be authenticated and decrypted by the TPM itself. Once TPM is adopted in a computer, another remote computer may request the TPM to provide an attestation report about the health status of of the computer. If the report signed by the TPM proves that the computer has the same health status the remote computer knew, the computer is trustable. There is no way that a malware can forge the report since that the only way to modify health status register is to extended-hash that only the TPM chip has the private key to sign the report. The extended hash is achieved by first concatenating the value in register and the newly arrived value and then hashing the concatenated value into the register.

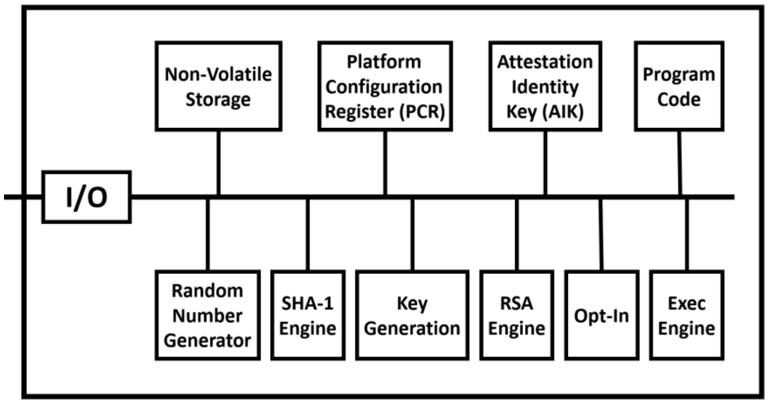


Figure 1: The architecture of the TPM1.2

IBM has implemented a software version of TPM, the sTPM, on Linux to facilitate the development of TPM-based systems and environment. sTPM can be used as a prototype to test software compatibility in the development stage. For those who want to understand the principles of TPM and develop TPM applications, the use of sTPM makes development work easier. For example, one can use Wireshark to probe TPM byte-stream so that he can diagnose the correctness of the operands of the command he sent. Running sTPM on a modern micro-system, such as Raspberry Pi(RPi) rather than on a desktop PC is more cost-effective for development purposes. In fact, sTPM for TPM 1.2 has been ported to Raspberry Pi that running Raspbian Wheezy operating system by Marcus et al. [1]. sTPM, however, has no non-volatile storage. This means that it will enter an initial state when started up.

In their porting work of porting sTPM a software stack, the TrouSerS, is used to facilitate application programs on a desktop computer to communicate with the sTPM (Figure 2). In figure 2, the client mimics a host computer while the Raspberry Pi mimics a TPM chip. Each TPM command sent is first marshalled into a byte stream and sent to RPi through Ethernet interface using TCP/IP network stack. TrouSerS also provides a command line interface so that one can type ‘tpm_takeownship’ command to take ownership over the sTPM and then test the functions of the sTPM.

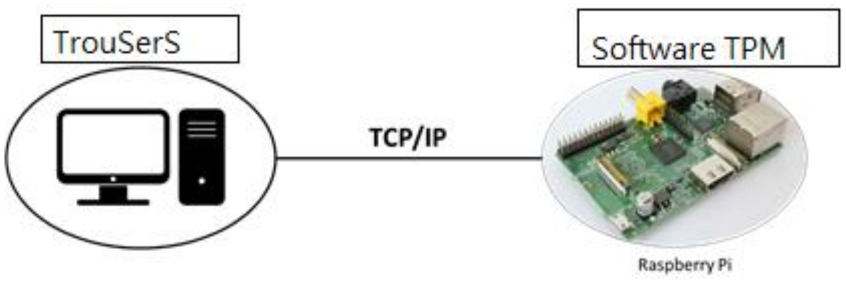


Figure 2 :Software TPM 1.2 connected with TrouSerS [1].

The first TPM specification released is TPM1.2, which specifies only RSA and SHA-1 as its core encrypting and hashing algorithm respectively [2]. Nowadays, new algorithms such as SHA-256, SHA-384, SHA-512, and ECC are developed to improve either efficiency or strength of security. Therefore, trusted computing group thus released new TPM technology (TPM2.0) on 2016 [3]. TPM2.0 provides enhanced features as follows: three persistent hierarchies (endorsement, platform and storage hierarchies) rather than one(endorsement hierarchy) are used, various key derivation functions (KDFs) are provided, and more PCR banks associated with various hashing algorithms are added. Besides, TPM2.0 supports advanced features such as direct anonymous attestation (DAA), privacy CA, and Extended Authorization (EA). All these features intend to widen the applications of TPM while provide the elasticity for computer manufacturers and system managers.

Given the complexity of TPM2’s functionality and that it is not compatibility with TPM1.2, new software stack, the TPM2 Software Stack (TSS2) and a command interpreter, TPM2_TOOLS to deal with the complicated functions and their arguments. The former translates each TPM2 command into a byte stream in a format called TPM command transmission interface (TCTI) while the latter provides command line interface and APIs to ease the use of the TPM commands. The specification of TCTI is well documented in TPM2.0’s specification. If one properly configures TPM2_TOOLS, the byte stream of TPM commands can be sent to one of the interfaces including network (TCP/IP), SPI, or local sTPM.

Given the popularity and complication of TPM2.0, there is a need of monitoring its behaviour while developing secure systems based on TPM. Thus we installed sTPM2.0 on RPi2, which is a faster version of Rpi and then evaluate its performance.

2. INSTALLATION METHOD (TPM2.0、TSS2、TOOLS)

The architecture of our work is shown in Figure 3. TSS2 and TPM2_TOOLs were installed on a desktop computer while the software TPM2.0(sTPM2.0) was installed on Rpi2. The operating system running on the former computer is Ubuntu 16.04 while that of the latter is Ubuntu mate 16.04. As of our experiences, installing TSS2 and TPM2-TOOLS on other Linux operating system is easy. Though many operating systems can be installed Rpi2 [4], only a few of them can used as the platforms of sTPM2.0 according to our testing result. For one wants to use another operating system on which sTPM2.0 is to be installed the OpenSSL library of the system must be version 1.0.

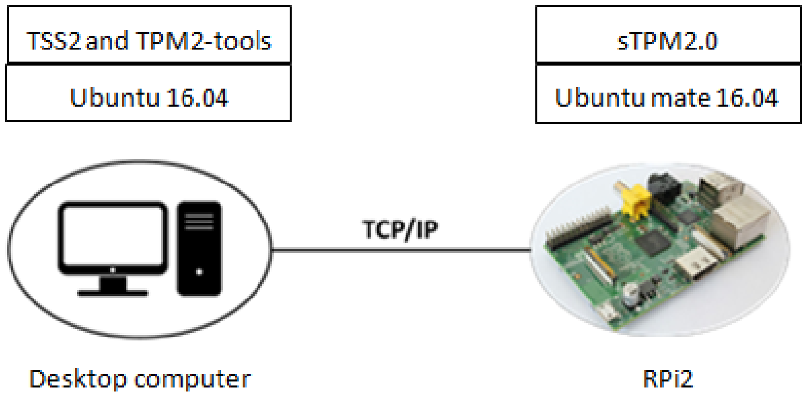


Figure 3: Architecture of sTPM2.0 emulator

The kits to install TSS2 and TPM2-tools can be found on GitHub [5]; sTPM2.0 source can be found on IBM's website[6]. Both computers must install the same supporting package for TPM2.0 before installing above software. The next three subsections describe the procedure to install the package, sTPM2.0 (on Rpi2), TSS2 and TPM2-tools(on Desktop computer).

2.1. Installation of TPM2.0 Supporting Packages

TPM2.0 software (TSS2, TPM2-TOOLS, and sTPM2.0) needs twenty-two packages to work properly on Linux operating system. Before install the packages, the operating systems on the desktop computer and Rpi2 must be patched using 'apt update' and 'apt upgrade' commands. The packages are then installed using following commands:

```
sudo apt-get -y install \autoconf  \autoconf-archive \automake
                                        \build-essential \doxygen \g++ \gcc \git
                                        \iproute2 \libcmocka0 \libcmocka-dev
                                        \libcurl4-openssl-dev \libdbus-1-dev
                                        \libgcrypt20-dev \libssl-dev \libtool \m4
                                        \net-tools \pkg-config \procps \uthash-dev
sudo apt -y update
```

2.2. Installation of sTPM2.0

A few steps are required to install sTPM2.0. The first step is to download the source from website provided by IBM [6]. Following are the commands to install it.

```
wget https://downloads.sourceforge.net/project/ibmswtpm2/ibmtpm532.tar
mkdir ibmtpm532
tar axf ibmtpm532.tar -C ibmtpm532
make "CC=gcc -Wno-tautological-compare" -C ibmtpm532/src
```

To start executing sTPM2.0 in background mode one can type in following command:

```
./ibmtpm532/src/tpm_server &
```

The sTPM2.0 will respond a TCP port number to and from which the command byte stream and result can be sent and received respectively. In our work the port number obtained is 2321.

2.3. Installation of TSS2 and TPM2-TOOLS

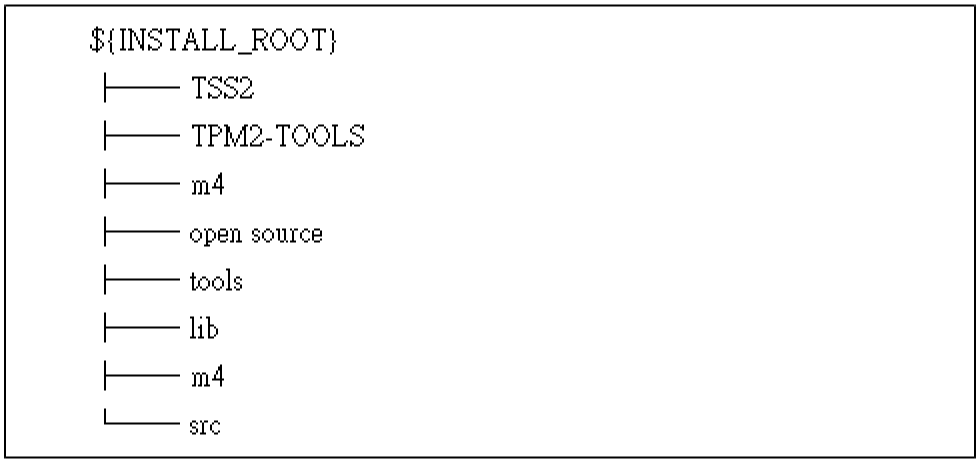
The suggested installation procedure of TPM2-TOOLS can be found in GitHub[5]. However, the lacking of macro definition errors found during the compilation process when we followed the procedure. To correctly compile TSS2 and TPM2-TOOLS on the underlying operating system, one must explore the location and the version of the Autoconf-archive of his operating system. In our case, following commands worked well on installing TSS2 and TPM2-TOOL on Ubuntu 16.04 operating system:

```
wget http://ftpmirror.gnu.org/autoconf-archive/autoconf-archive-2017.09.28.tar.xz
tar axf autoconf-archive-2017.09.28.tar.xz
cp autoconf-archive-2017.09.28/m4/ax_code_coverage.m4 m4/
./bootstrap && ./configure --prefix=$HOME && make -j4
make install DESTDIR=$HOME/my-tpm2-tss-installation-dir
sudo cp --symbolic-link -r $HOME/my-tpm2-tss-installation-dir/* /
sudo ldconfig
```

Installing TPM2-TOOLS is similarly as below:

```
cp -R ~/tpm2-tss/m4 ~/tpm2-tools/m4
./bootstrap && ./configure --prefix=$HOME && make -j4
make install DESTDIR=$HOME/my-tpm2-tss-installation-dir
```

After the installation, the file structure can be seen as Figure 4.



```
${INSTALL_ROOT}
 ├──── TSS2
 ├──── TPM2-TOOLS
 ├──── m4
 ├──── open source
 ├──── tools
 ├──── lib
 ├──── m4
 └──── src
```

Figure 4: Directory structure of TSS2 and TPM2-TOOLS

2.4. Connecting to sTPM2.0

Environment variables of TSS2 or TPM2-TOOLS must be configured properly so that the command byte stream to sTPM2.0 on Rpi2. Remember that the port number we obtained in section 2.2 is 2321. Assume that the IP address of RPi2 is 192.168.11.11 then following commands set the configuration we desired:

```
export _TPM2TOOLS\_TCTI\_NAME_="mssim:tcp:// 192.168.11.11:2321"
export _TPM2TOOLS\_TCTI_="mssim:tcp:// 192.168.11.11:2321"
export _TPM2TOOLS\_TCTI_="mssim:host=192.168.11.11,port=2321"
export TPM2TOOLS_TCTI="mssim:host=192.168.11.11,port=2321"
cd $HOME/my-tpm2-tss-installation-dir/root/bin
./tpm2_startup --clear --tcti mssim:host=192.168.11.11,port=2321
```

Then we can start to test functions of TPM2.0.

3. TESTING TPM COMMANDS

Key generating function is one of the important functions of TPM. In following three steps, we create an RSA key pair for endorsement. The detail of the commands can be found in tutorial of TPM2-TOOLS [7].

Step 1: Create a primary key for Endorsement Hierarchy.

The primary key acts as the key ring with which children keys can be generated and managed. The command to create a primary key is as followed, which return a context file associated with the primary key:

```
./tpm2_createprimary -a e -g 0x000B -G 0x1 -c primary_ctx
```

Step 2: Generate a children key pair (public and private keys) beneath the primary key.

Following command asks TPM to generate and then return an RSA key pair, said key_pub and key_priv of 2048 bits designated by the option -G. The key pair is protected by the primary key.

```
./tpm2_create -g 0x000B -G rsa2048:rsaes -C primary_ctx -u key_pub –r key_priv
```

STEP 3: Load the key pair into TPM.

TPM has very limited space to store keys. Thus the key pair in above step is not stored in the TPM. To use the key pair, one must load it into the TPM as followed:

```
./tpm2_load -C primary_ctx -u key_pub -r key_priv -c rsa_key
```

Next, we tested sTPM2.0's advanced functions: protecting a secret , encrypting a file, signing a message, and verifying the signature, all using above key pair for purpose of example illustration. For more detail, please refer to the tutorial of ManKier [8].

STEP 4: Create free-form objects for "sealing" small amounts of data to the TPM

Following command with option -I- asks TPM2.0 to create an object for sealing the string 'secret' from stdin. The password '12345' is used to protect the secret and file obj_pub and obj_priv is respective the public portion and private portion of the created object.

```
echo "secret" | ./tpm2_create -C primary_ctx -I- -p 12345 -u obj_pub -r obj_priv
```
Following command asks TPM2.0 to get the sealed data protected by primary key.
```
./tpm2_load -C primary_ctx -u obj_pub -r obj_priv -o seal.ctx
```
To retrieve the secret out of the TPM, one must use password '12345' in following command.
```
./tpm2_unseal -c seal.ctx –p 12345
```

Notice that sTPM2.0 does not have non-volatile storage, thus a new primary_ctx will be generated if one restarted sTPM2.0 and re-created an endorsement hierarchy. Thus the secret protected by the old primary_ctx cannot be recovered. When using sTPM2.0, one should back up the secret in other place by himself. The TPM2.0 chip, on the contrary, has built-in non-volatile memory to retain the secret after power down.

STEP 5: Encrypting a file using the loaded key.

Following command ask TPM2.0 to encrypt data in file named input_data. The encrypted data is then stored as a file named encrypted.out. Here the reading and writing operations of the files are achieved by TPM2-TOOLS.

```
./tpm2_rsaencrypt -c rsa_key -o encrypted.out input_data
```
To decrypt the encrypted data in a file, type following command:
```
./tpm2_rsadecrypt -c rsa_key -o plain.out encrypted.out
```

STEP 6: Signing a message with designated hashing algorithm.

Following command ask TPM2.0 to sign a message with SHA256 hashing algorithm. The signature is put in the file 'sig.rssa.'

```
./tpm2_sign -c rsa_key -G sha256 -m message.dat –o sig.rssa
```

To verify the signature, one needs to load the public key associated with the signature first and then verify the signature as below.

```
./tpm2_loadexternal -C o -Grsa -u Key_pub -c key.ctx
./tpm2_verifysignature -c key.ctx -G sha256 -m message.dat -s sig.rssa
```

The tpm2_loadexternal command can be omitted provided the public key is already in the TPM.

4. THE PERFORMANCE TEST

The goal of our work is that one can use sTPM2.0 on RPi2. Functions can be verified through the API and command line interface provided by TSS2 and TPM2-TOOLS. Yet the performance of sTPM2.0 on RPi2 is another metric concerned by TPM application developers. In this section, we measure the performance of sTPM2.0 on RPi2 and compare it to performance results previously measured.

4.1. Previous Performance Measurement Results

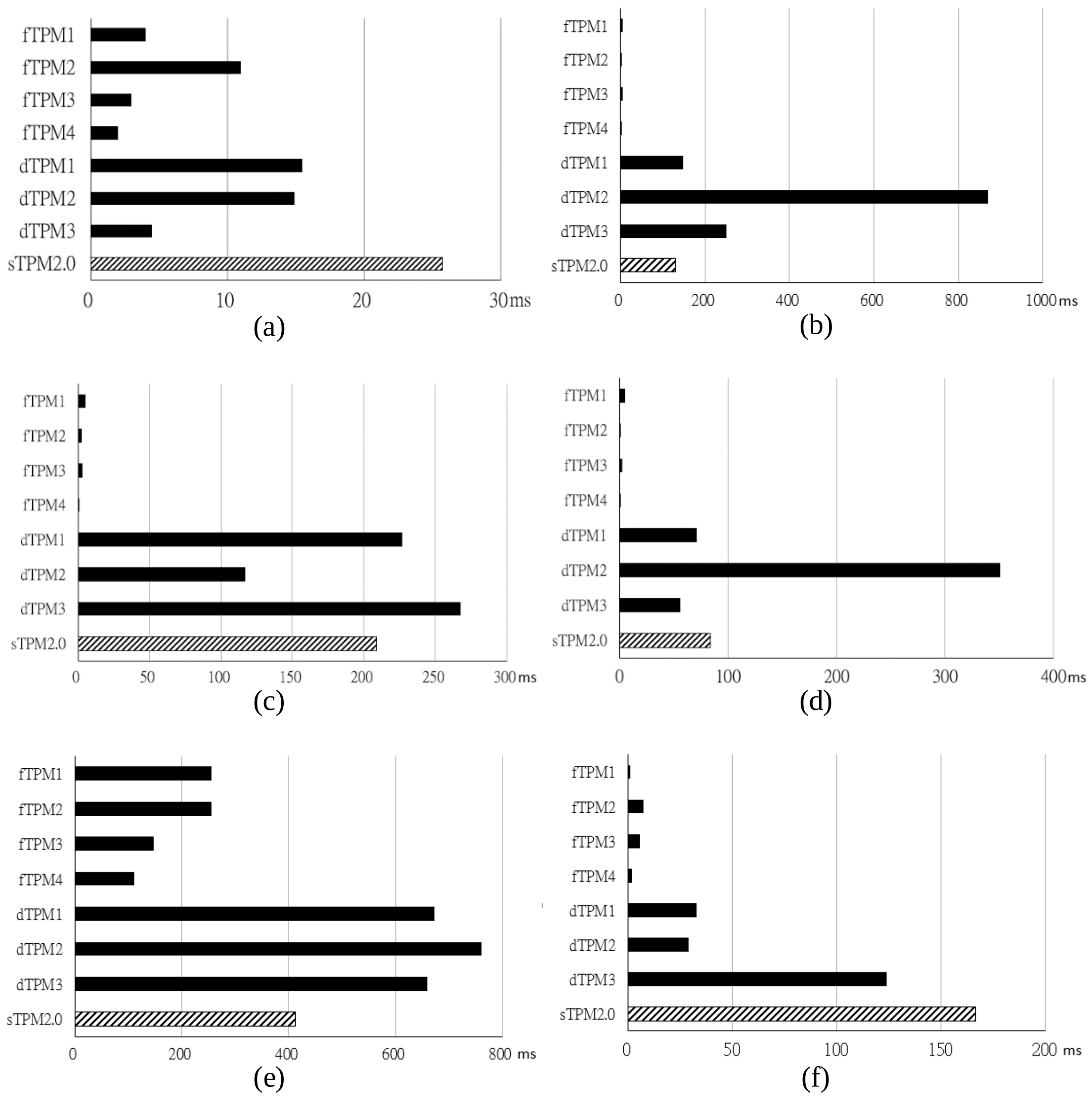
Eight TPM benchmark commands are selected for performance test: (a) Creating an RSA Key; (b) Loading key; (c) Sealing; (d) Unsealing; (e) Signing; (f)Signature verification; (g) Encryption; (h) Decryption. Each of these commands is sent to sTPM2.0 and measured. To measure the execution time of each command sent to sTPM2.0, we modify its source code so that the tick count between the command received and the response sent is recorded. The performance is compared with seven other TPM devices, as listed in Table 2. Among them, fTPM1~fTPM4 are four commercially available off-the-shelf mobile devices equipped with fTPM, and their processor models are provided. dTPM1-3 are commercially available TPM chips, however their processor type and frequency are kept confidential. RPi2 used in this study is listed in the bottom.

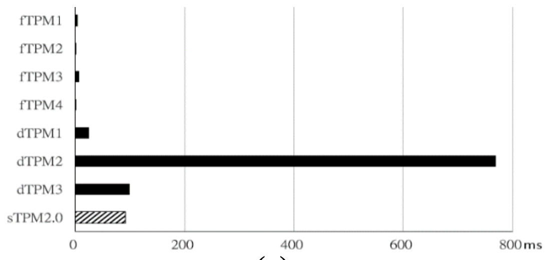
Table 1 The frequency and processor type ofvarious TPM2.0 devices

Name	Processing unit
fTPM1	1.2 GHz Cortex-A7
fTPM2	1.3 GHz Cortex-A9
fTPM3	2 GHz Cortex-A57
fTPM4	2.2 GHz Cortex-A57
dTPM1	No offer

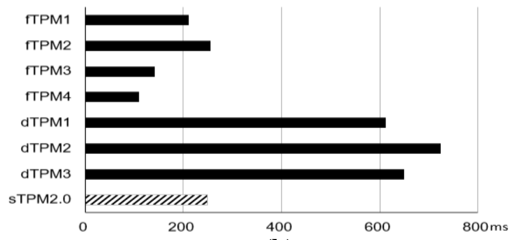
dTPM2	No offer
dTPM3	No offer
sTPM2.0	0.9 GHz Cortex-A

In each sub-figure of Figure 5, the performance of the TPM devices other than sTPM2.0 is quoted from previous report directly [9]. The x-axis in the figure indicate the time(in milisecond) required to finish a command.





(g)



(h)

Figure 5. Performance comparison among various TPM devices when executing TPM commands: (a) Creating an RSA Key; (b) Loading key; (c) Sealing; (d) Unsealing; (e) Signning; (f) Signature verification; (g) Encryption; (h) Decryption.

Several interesting observations are made. First, sTPM2.0 is slowest in creating the RSA key. The reason is that other TPM2.0 devices generate key pairs and store them in a queue before a key pair generation command is issued. sTPM2.0 does not leverage this technique. Secondly, the performance of fTPM and dTPM on seal and unseal commands are huge. The reason is that the former fabricated a dedicated area on CPU chip while the latter use serial peripheral interface for communication, i.e. the bandwidth of communication channel affects the performance. Thirdly, the performance of sTPM2.0 on other commands is better than commercial TPM2.0 chips. Obviously, processors' computing power dominates the performance. The performance of fTPMs is highest based on the same reason. More detail implementation and performance comparison between fTPM and dTPM can be found in document written by Microsoft [9].

5. SUMMARY AND CONCLUSION

The specification of TPM2.0, and thus all TPM2.0 devices are still evolving. Not all commands in the TPM2.0 specification are implemented in the TSS2 and TPM2-TOOLS. For example, command 'pcrlist' had been replaced by command 'pcrread'. Another example is that only three handles can be imported to sTPM2.0, which can easily result in inadequate space for objects used by commands. Though TSS2 has RM (Resource Manager) to solve the problem of inadequate handler space, it is not yet implemented in TSS2 at the time we downloaded it. Not even the command to flush handle in sTPM2.0 is implemented. The flush command is available till we were testing advanced functions. We suggest one to obtain the latest version of all software for sTPM2.0 to more closely meet the specification of TPM2.0.

Nonetheless, one can sTPM2.0 for prototyping his system that leverages TPM2.0 technology provided that he saves the transient sealed secret into non-volatile location.

Trust computing forms the basis of Internet security. Many applications started to use TPM techniques to protect data, e.g. Bitlocker, vehicle firmware upgradation, etc. This study reveals how to set up a sTPM2.0 on Rpi2 and reveal the performance to facilitate one to develop security applications based on TPM2.0.

REFERENCES

[1] E. N. Marcus Sundberg, "Emulation of TPM on Raspberry Pi," Lund University Publications Student Papers, Lund, Sweden, 2015.

[2] T. C. Group(TCG), "TPM 1.2 Main Specification," Trusted Computing Group(TCG), 2 10 2003. [Online]. Available: https://trustedcomputinggroup.org/resource/tpm-main-specification/. [Accessed 30 7 2018].
[3] Trusted Computing Group, "TPM Library Specification 2.0," 1 October 2014. [Online]. Available: https://trustedcomputinggroup.org/resource/tpm-library-specification/. [Accessed 21 April 2018].
[4] T. R. P. Foundation, "Downloads," The Raspberry Pi Foundation, [Online]. Available: https://www.raspberrypi.org/downloads/.
[5] W. Roberts, "Getting Started • tpm2-software/tpm2-tools Wiki • GitHub," 24 Jul 2018. [Online]. Available: https://github.com/tpm2-software/tpm2-tools/wiki/Getting-Started.
[6] IBM, "IBM's Software TPM 2.0 download | SourceForge.net," IBM, [Online]. Available: https://sourceforge.net/projects/ibmswtpm2/.
[7] W. Roberts, "tpm2-software/tpm2-tools Creating Objects," [Online]. Available: https://github.com/tpm2-software/tpm2-tools/wiki/Creating-Objects.
[8] [Online]. Available: https://www.mankier.com/package/tpm2-tools.
[9] S. S. H. Raj, "fTPM: A Firmware-based TPM 2.0 Implementation," Microsoft Research, pp. 0-23, 17 11 2015.

AUTHORS

Jen-Ling Cheng got his Bachelor, Master, and Ph.D. on 1985, 1987, and 1994 respectively, all from department of computer science and information engineering, National Chiao-Tung University, Taiwan. He is currently an associate professor of department of medical informatics, Tzu-Chi University, Taiwan.

Kuang-Chi Chen got her Ph.D. in 2004 with major at Biostatistics from Institute of Epidemiology, National Taiwan University, Taiwan. She is currently an associate professor of department of Medical Informatics, Tzu-Chi University, Taiwan.

Hong-Wei Zhang was born in Taoyuan, Taiwan, on November 30, 1995. He received his bachelor degree from department of medical informatics, Tzu-Chi University, Taiwan. Currently he is doing master degree in the same department. His research interests are cryptography and artificial intelligence.

Wei-Yu Chen graduated from department of Medical Informatics, Tzu-Chi University on 2019. He is currently working for his Master degree in the same department.

Dai-Ye Wu was born in Hualien, Taiwan, on September 25, 1997. She received her bachelor degree from department of medical informatics, Tzu-Chi University, Taiwan. Currently she is doing master degree in the same department. Her research interest is in cryptographic IC design.