Jurnal Teknologi

Full paper

Towards a New Framework for TPM Compliance Testing

Usama Tharwat Elhagari[a*], Bharanidharan Shanmugam[b], Jamalul-lail Ab. Manan[c]

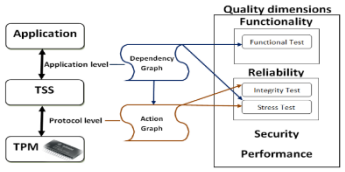
[a]*Faculty of Computing, Universiti Teknologi Malaysia, 81310 UTM Johor Bahru, Johor, Malaysia*
[b]*Advanced Informatics School, 81310 UTM Johor Bahru, Johor, Malaysia*
[c]*MIMOS Berhad, Malaysia*

*Corresponding author: elhagari_u@yahoo.com

Article history

Received :10 December 2014
Received in revised form :
1 February 2015
Accepted :12 February 2015

Graphical abstract



Abstract

Trusted Computing Group (TCG) has proposed the Trusted Computing (TC) concept. Subsequently, TC becomes a common base for many new computing platforms, called Trusted Platform (TP) architecture (hardware and software) that, practically, has a built-in trusted hardware component mounted at the hardware layer and a corresponding trusted software component installed at the operating system level. The trusted hardware component is called Trusted Platform Module (TPM) whose specification has been issued by TCG group and it is implemented by the industry as a tamper-resistant integrated circuit. In practice, the security of an IT TPM-enabled system relies on the correctness of its mounted TPM. Thus, TPM testing is urgently needed to assist in building confidence of the users on the security functionality provided by the TPM. This paper presents the state of the art of the modelling methods being used in the TPM compliance testing as well as it demonstrates some of the important attacks against TPM. Finally, the paper proposes new framework criteria for TPM Testing that aim at increasing the quality of TPM testing.

Keywords: Trusted platform module; compliance testing; modelling; trusted computing; FSM; EFSM

1.0 INTRODUCTION

Recently, software on computing platforms has become increasingly complex leading to a large number of potential vulnerabilities. Consequently, protecting information technology systems through software-based mechanisms has become increasingly more unable to solve all security problems there in. To mitigate this issue, hardware-based embedded security solutions have been used in the information technology industry. Among the key advances, Trusted Computing Platform Alliance (TCPA), which was later replaced by the Trusted Computing Group (TCG), proposed the Trusted Computing (TC) concept. Subsequently, TC became the common base for many new computing platforms, called Trusted Platform (TP) architecture that, practically, has a built-in trusted hardware component at the physical level and corresponding trusted software component at operating system level. The trusted hardware component is called Trusted Platform Module (TPM) whose specification was issued by the TCG group and is implemented by industry as a tamper-resistant integrated circuit. TPM is dedicated to performing cryptographic functionality and to securely store cryptographic keys and secrets.

Since the last couple of years, hundreds of millions of PC laptops and desktops have been equipped with TPM chips. In fact, there are many different vendors that produce TPM chips, such as Atmel, Infineon, Broadcom, Sinosun and STMicroelectronics/Winond, and, of course, with different modes of implementation. This implies that there is an urgent need to have a testing methodology that can help security application developers and end-users to verify the compliance of their TPM-enabled systems with respect to TCG specifications.[1,2]

Past research works in the area of TPM testing fall into two broad categories, namely; compliance testing,[2-8] and security analysis on the TPM specifications,[2,9-15]. This paper presents several modelling methods which are in the domain of TPM compliance testing. Recent efforts show that many TPMs available in the market are non-compliant to the TCG specification.[2-8]. At this point, it is worth mentioning that China has its own specification and its trusted hardware component is called Trusted Cryptography Module (TCM). The TCM chip has been specified and manufactured by China. It was concluded that there was a gap between the TCM implementations and the Chinese specification.[16] This paper also presents the state of the art of some important attacks that have been conducted against the TPM during last years. We begin with the modelling methods of TPM specifications in section 2. Sub-section 2.1 is the discussion on the informal method of TPM compliance testing (with an example). Modelling of TPM specification based on FSM and EFSM (with examples) are presented in sub-section 2.2 and sub-section 2.3 respectively. Section 3 presents the attacks against TPM. This paper is concluded with proposing features of a new framework for TPM testing in section 4.

■2.0 MODELLING METHODS OF TPM SPECIFICATIONS

There are mainly three methods that have been used in modelling the TPM specifications. The TPM testing was first introduced using informal method.[1,17] On the other hand, next research efforts in TPM testing used two formal methods that are based on state machine theory namely, Finite State Machine (FSM) and Extended Finite State Machine (EFSM).[4,5,16] In the next sub-sections a brief discussion on the following three methods; informal modelling, FSM-based modelling and EFSM-based modelling is presented.

2.1 Informal Modelling

The TPM compliance testing was first introduced using informal method in which TPMs from different vendors were evaluated.[1,17] In informal modelling, testing is conducted in two levels and two quality dimensions, as shown in Figure 1.

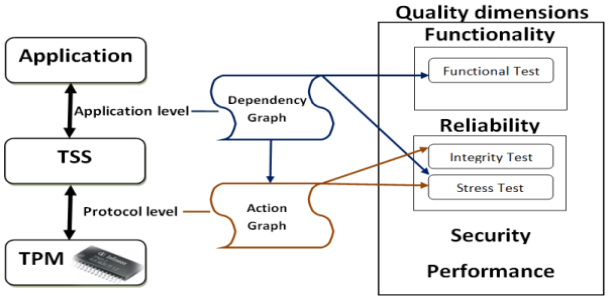


Figure 1 Compliance testing levels and quality dimensions of TPM

Firstly, the Compliance Testing Levels consists of the application level and the protocol level. At the application level, the TPM is tested from real application standpoint to test the TPM functionality. The protocol level is dedicated to test the TPM's commands with respect to the data structures. Secondly, in Compliance TPM Testing there are four core quality dimensions namely, functionality, reliability, security and performance. Nevertheless the conducted informal method, only two quality dimensions were considered,[1,2] which are functionality and reliability. Notably, under the functionality dimension, only a function test is conducted. Whereas under the reliability dimension, integrity test and stress test are conducted. In this paper, the other quality dimensions namely, security and performance are discussed in later subsections.

Two other aspects of the Compliance TPM testing include, "TPM behavior", which is examined via function test and "TPM behavior upon failures" which is examined using the integrity tests. Yet another aspect is the stress tests which examine "TPM behavior under extreme conditions".

Here, we emphasize and focus on their method of generating test cases to test the data structure of TPM's commands. In order to test the data structure of a single command, many test cases are needed to test the command parameters. Thus to generate test cases for each command, the command execution is modeled as a state transition into a return code, as shown in Figure 2.

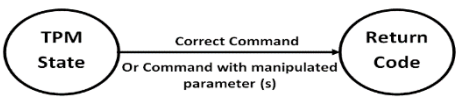


Figure 2 TPM command execution model

It is observed that the number of test cases in compliance testing is an issue. To mitigate this issue, the input parameters of the TPM commands are categorized into four different categories based on input parameters which are described below[1,17]:

Valid: they are acceptable inputs and allow TPM to correctly and successfully process the command. Consequently, the return code must be TPM_SUCCESS.
The following three categories should return code indicating an error:
Illegal (A): these are inacceptable inputs as they have either wrong data structure or unspecified values, which are not stated by TPM specifications,
Invalid (B): these are inacceptable inputs as their values are wrong or meaningless values.
Unsupported (C): these are inputs with values stated by TPM specifications but not acceptable in the context of the command.

The steps of the integrity tests at the protocol level are as follows[1,17]:

(1) Study in detail the TPM specifications.
(2) Categorize the TPM commands based on their related TPM functionality (Dependency Graph).
(3) From the Dependency Graph draw the action graph which shows the required execution order of the TPM commands for successful individual TPM commands execution.
(4) Define the state(s) at which the command (under test) is allowed to execute.
(5) Define the TPM return code(s) for those state(s) at which the command is not allowed to execute.
(6) Construct a table/graph showing all the command parameters after manipulation and the related return codes. Table 1 shows TPM_CreateWrapKey as an example.
(7) Execute all the commands required, indicated by the action graph, for the successful command execution.
(8) Send the command input message with only one manipulated parameter to the TPM.
(9) Compare the return code from the TPM with the expected one as stated in the table/graph.
(10) Repeat step 9 and 10 for each manipulated parameter.
(11) If all the return codes from the TPM match the expected ones then the implementation of the command under test is complaint with TPM specification, based on integrity test only.
(12) Repeat step 2 up to step 11 for TPM commands, stated on the TPM specifications.

The research work[1,17] that used the informal method is considered as the founder of TPM compliance testing and has contributed valuable knowledge and experience significantly in TPM testing. Based on the results of the conducted informal method we know that some TPM implementations which are from (Infineon, Atmel, and ST STM 19 WP 18) were found to be incompliant with TCG specification and have security related bugs. However, the method used in determining the compliance was still informal[16] and, furthermore its generation of test cases was not automatic and the test method needs to be reviewed and improved so that it becomes more systematic[2].

It is generally known that manual generation of test cases is an expensive, error-prone and time consuming process. Nowadays, with the improvement of TPM implementations, the informal method and manual generation of test cases might not be so effective in dealing with greater number of cases of incompliance of TPM implementations.

Table 1 TPM_CreateWrapKey command and its related TPM's return codes

STATE	Parameter name		Input Type	Return Code
S2,S4, S6,S8	**TPM_CreateWrapKey Input Message**			TPM_DISABLED
S3				TPM_DEACTIVATED
S5				TPM_NOSRK
S7				TPM_DEACTIVATED
				TPM_NOSRK
S1	tag		A	TPM_BADTAG
			B	
	paramSize		B	TPM_BAD_PARAM_SIZE
	ordinal		A	
			C	
	parentHandle		B	TPM_INVALID_AUTHHANDLE
				TPM_KEYNOTFOUND
			C	
	dataUsageAuth			
	dataMigrationAuth			
	keyInfo			
	ver			
	keyUsage		A	TPM_INVALID_KEYUSAGE
			C	
	keyFlags		A	TPM_BAD_PARAMETER
			C	
	authDataUsage		A	TPM_BAD_PARAMETER
			C	
	algorithmParms		A	TPM_BAD_KEY_PROPERTY
			C	
				TPM_NOTFIPS
		algorithmID		
	authHandle		B	TPM_AUTHFAIL
			C	TPM_INVALID_AUTHHANDLE
	authLastNonceEven			
	nonceOdd			
	continueAuthSession			
	pubAuth		B	TPM_AUTHFAIL

2.2 FSM-based Modelling Method

Mealy machines and Moore machines are two types of finite state machines or finite automata. These are widely used to model finite state systems in different areas such as communication protocols and sequential circuits.

Definition 1: a deterministic finite state machine (FSM) D is a six-tuple:

$D = (S, I, O, \delta, \lambda, s_{init})$ where S, I , and O are finite and non-empty sets of states, input alphabet and output alphabet, s_{init} is the initial state, δ: SxI→S and λ:SxI→O are the functions of state transition and output, respectively.

The conformance of system implementation to the system specification can be tested by using FSM. This problem is called conformance testing or fault detection problem;[3] at which two FSMs are given: a specification machine SPEC and implementation machine IMP. We can only observe the behavior of IMP that is a black box.

To test the conformance of an implementation under test IUT to its specification, it is needed to generate test cases from the SPEC model and then apply these test cases to the IUT. Test cases can be generated automatically from SPEC. A test case contains input and expected output. Therefore IUT conforms to its specification if it passes all the test cases.

TPM operational states, that are shown in Table 2, were modelled, and the commands of TPM based on deterministic finite state machine.[4,5] There are four FSM models have been constructed which include the TPM operational states, TPM disabled-command suite, TPM deactivated-command suite and TPM unowned-command suite.

Table 2 TPM operational states

State	Enable/Disable	Active/Inactive	Owned/Unowned
S1	Enable	Active	Owned
S2	Disable	Active	Owned
S3	Enable	Inactive	Owned
S4	Disable	Inactive	Owned
S5	Enable	Active	Unowned
S6	Disable	Active	Unowned
S7	Enable	Inactive	Unowned
S8	Disable	Inactive	Unowned

We give an explanatory example for modelling TPM specifications based on FSM; Figure 3 shows the FSM model of the eight TPM operational states. This example is based on the reported methodology.[4,5] The parameters of the FSM model are as follow:

$D_0 = (S_0, I_0, O_0, \delta_0, \lambda_0, sinit_0)$
$S_0 = \{s_1, s_2, s_3, s_4, s_5, s_6, s_7, s_8\}$
I_0 = {TPM_OwnerSetDisable, TPM_PhysicalDisable, TPM_PhysicalSetDeactivated, TPM_SetTempDeactivated, TPM_OwnerClear, TPM_ForceClear, TPM_PhysicalEnable, TPM_TakeOwnership}
O_0= {S} where S means that the TPM successfully has executed the related command.
$s_{init0} = s_5$
Bread-First Search has been used to generate test cases from D_0.

Basically, FSM is used to model the control portions of system specification. This could be the main weakness of FSM as system specification normally contains data dependencies between the specification parts; which means that FSM is not powerful enough to model concrete systems in a concise way.[3] Consequently, FSM model may have issues such as state explosion as the number of states increases rapidly[6] and FSM is not realistic in most practical situations.[7] According to the TPM specification, majority of TPM's commands are dependent on data from each other and a successful command execution may need other command(s) that have been successfully executed. Therefore, modeling the TPM specification using FSM, taking into account control and data dependencies between the commands, could result in impractically huge model and consequently having state explosion problem. Furthermore, the data dependency of the TPM's commands should be tested to determine the behaviors of the TPM implementation.

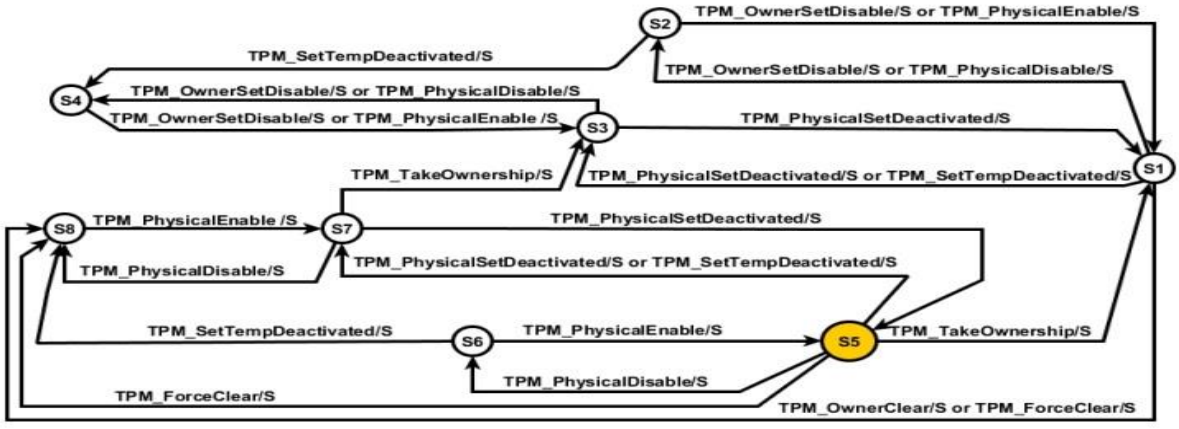


Figure 3 FSM model for the TPM's operational states

2.3 EFSM-based Modelling Method

EFSM[8] is generalization of FSM; i.e. EFSM is a traditional Mealy FSM extended with variables, predicates, and operations. Additionally, one main advantage of EFSM over FSM is that EFSM helps in reducing number of states. This advantage is because of the fact that EFSM is able to model the control flow of a system while its data flow is represented by variables, predicates, and operations.

Definition 2: An EFSM is a six-tuple [6, 9] (S, s_0, I, O, T, V) where S is a non-empty finite set of states, $s_0 \in S$ is the initial state, I and O are non-empty finite sets of input and output interactions, T is a non-empty finite set of transitions and V is a non-empty finite set of variables. $t \in T$ is a six-tuple (s_i, s_e, x, c, y) where s_i, $s_e \in S$ denote the initial and terminating states of t, respectively, $x \in I$ is the input interaction of t, c is a logical expression representing a condition of t and expressed in terms of the variable of V, $y \in O$ is the output interaction of t.

EFSM-Based specification modelling was used in trusted computing[16], where it is reported that the specifications of the Trusted Cryptography Module (TCM) were modelled by using EFSM. Firstly, the dependencies between the TCM commands were de-fined and, consequently, a dependency graph was drawn. Secondly, an EFSM model was constructed and test cases were generated for the EFSM model. The authors mentioned that the test case generation was not fully automatic. Finally, the TCM compliance testing was conducted in two layers, namely: command-level and function level. The former was used to test the TCM reliability, i.e. its behaviour when receive legal-manipulated command message, as well as testing the TCM robustness where the behaviour of the TCM was tested by sending illegal-manipulated command message. In the latter, functionality test was conducted for testing the TCM functions.

To give an illustrative example of the EFSM modelling of the TPM specifications, Figure 4 shows EFSM model for a portion of the TPM specification, storage functions sub-module and some commands of the admin ownership module sub-module. This example adopts the reported methodology. [16] The EFSM model was constructed based on the research work of and the TPM specification version 1.2, level 2 revision 116. As can be seen from Figure 4, the parameters of the EFSM model are as following:

S= {S_1, S_2, S_3, S_4};
s_0= S_1;
I={ TPM_TakeOwnership, TPM_OwnerClear, TPM_ForceClear, TPM_DisableOwnerClear, TPM_DisableForceClear, TPM_Seal, TPM_Unseal, TPM_Unbind, TPM_CreateWrapKey, TPM_LoadKey2, TPM_GetPubKey, TPM_Sealx}
O= {Create Owner, Clear Owner, Create Key, Disable ForceClear, Disable Owner-Clear, Load Key, Unseal, Seal, UnBind, Get PubKey}
V= {Ownership Enabled, KeyLoaded, KeyExists, OwnerClearEnabled, ForceClearEnabled}
There are 13 transitions where t1∈T is TPM_TakeOwnership [OwnershipEnabled]/ Create Owner.

The EFSM-Based specification modelling[9] has made some improvement to the FSM-based modelling[4,5] in modelling and generating test cases. However, it lacks the automatic generation of test cases. Furthermore, in order to use this method in TPM compliance testing it needs to involve the internal TPM data, such as flags, as variables to represent the relationship among the TPM commands.

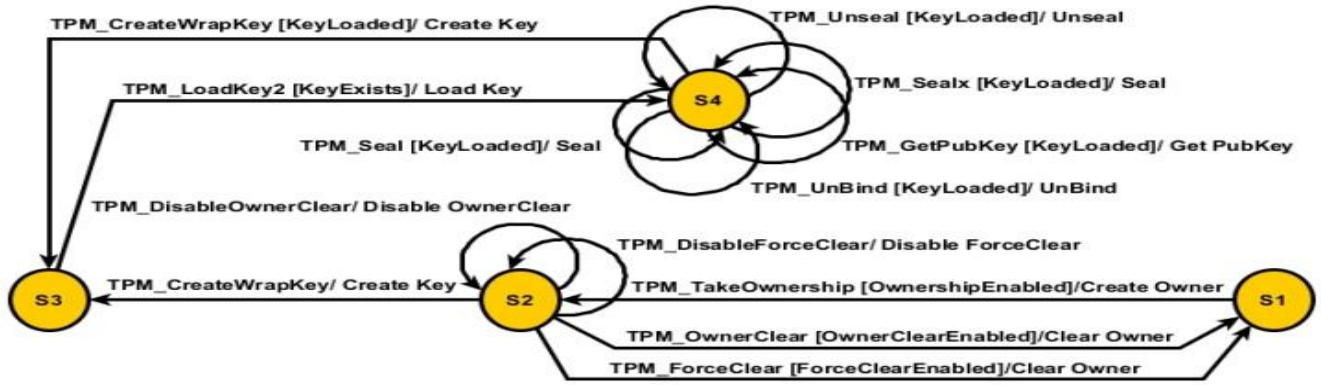


Figure 4 EFSM model for the storage functions and admin ownership sub-modules

3.0 ATTACKS AGAINST TPM

Although the main function of TPM chips is establishing trust and is to provide security services to their host platforms, many attacks have been performed against either the TPM chip itself or its environment, such as communication interface with the other platform's components. These attacks are either practical attack or security flaws that have been revealed by security analysis research work on the TPM specifications. Table 3 shows a collection of attacks against TPM and LPC bus (communication interface with other TP's components).

Table 3 Attacks against the TPM by year

Attack	Year of the Attack								
	2004	2005	2006	2007	2008	2009	2010	2011	2013
Physical Attack							X		
Attacking TPM-Based architecture									
Against TXT					X				
Against BitLocker						X			
Dictionary Attack									
Online		X	X						
Offline					X				
Replay Attack		X				X			
Attacking the TPM Communication Interface (LPC Bus)									
Passive		X							
Reset	X								
Violating the integrity of TPM commands				X				X	X

The attacks, as shown in Table 3, vary from simple attacks to sophisticated attacks. For instance, in the reset attack,[10] an attacker uses a small piece of wire in order to reset the TPM without resetting the whole TP. In other words, due to this attack the integrity values that represent the TP configuration and stored in TPM are changed to zeros. This violates the first design goal of TPM and breaks the remote attestation and sealing feature provided by the TPM. The passive attack is similar to the reset attack. Attackers can use inexpensive equipment to eavesdrop critical information from the LPC Bus.[11]

The Object-Independent Authorization Protocol (OIAP) is a TPM security protocol mainly intended to prevent replay attack. However, it was proved formally that OIAP has problem in its design which makes it vulnerable to replay attack.[12,13] Additionally, it is reported that the research work showed

formally an improper implementation of the OIAP which may lead to replay attack as well.[13]

TCG specification stated countermeasures against dictionary attacks, so TPM implementations contain protection mechanisms against dictionary attacks. Despite this protection, that mechanism was defeated.[1] Furthermore, in certain circumstances offline dictionary attack against TPM is possible which may lead to other issues, for example, "to impersonate the TPM owner to the TPM, or the TPM to its owner".[14]

As a result of formal analysis on the TPM specifications, versions 1.1 and 1.2, it is reported that the integrity of the TPM_CertifyKey command can be violated due to a design problem in the Hash-Based Message Authentication Code (HMAC) calculation.[15-17]

The most sophisticated attack against TPM, so far, is the physical attack which was performed by Christopher Tranovsky.[18] He was able to access TPM chips, by using electron microscope, from inside reaching the TPM data bus. So he was able to get any piece of information stored in the chip, such as cryptography keys. This means that the tamper-resist feature of TPM has been defeated. A worst case scenario could be, "not only is the data on individual chips at risk from this attack, once the manufacture's code is copied from the chip it could be used to produce counterfeit chips, which also could contain backdoors".[19]

In addition to the above mentioned attacks, both of BitLocker, an encryption feature provided by Microsoft Windows, and the Intel Trusted Execution Technology (TXT) have been successfully attacked by Fraunhofer Institute for Information (SIT)[20] and Invisible Things Lab (ITL)[21] respectively.

Although the results of the security analysis on the TPM specifications do play a crucial role in evaluating the quality of the TPM specifications and subsequently the security functionality provided by the TPM chips that implemented based on the specifications, to the best of our knowledge, none of the existing TPM testing frameworks[1,4,5,22,23] has ever used these analysis results to evaluate the TPM under test.

4.0 CONCLUSION

Trusted computing (TC) is a promising technology for enhancing the security of computer systems and networks. TCG issued specifications for TC technology which is called TCG specifications. We emphasize on the TPM specifications. Based on past works it is discovered that there is a gap between some TPM implementations and the TPM specifications. This gap may cause the TPM component to fail in performing its security functionality and consequently may result in failing the security of its mounted system. Therefore, there is an urgent need to test the compliance of TPM implementation with reference to its specifications. In this paper, we report on some progress of the research works in the field of TPM testing have been achieved. The two major contributions of our work are on TPM compliance testing and security analysis on TPM specifications. In compliance testing of TPM, we presented the three modelling methods, namely, informal, FSM and EFSM. The main problem of these three methods is that there is a high possibility that it might cause state space explosion. Furthermore, the existing TPM compliance testing framework that we have referred to in the literature so far, conducted their tests based on test cases pre-generated earlier. In other words, a complete test suite must first be derived completely before conducting the TPM compliance testing. This approach is referred to as batch-mode testing. Additionally, to our knowledge, none of the existing TPM testing frameworks has ever used the results of the TPM security analysis to evaluate the TPM implementations.

We can safely conclude that testing security devices such as TPM needs to be done systematically through automatically generated random test cases to increase the quality of testing. Moreover, automatic security testing has never been emphasized as a quality dimension in the exiting Framework for TPM Testing. We have discussed and highlighted the urgent need to enhance the current TPM testing frameworks to achieve higher quality TPM testing.

For future work, we propose a new framework for TPM Testing that has several features. Firstly, it should have capacity to generate random test cases on-the-fly. This helps in alleviating the state space explosion problem and improves the quality of testing.

Secondly, it should posse other quality dimensions such as automatic security testing. Furthermore, it should be suitable for the TPM stakeholders such as normal TPM users who have abstract knowledge about TPM.

References

[1] Ahmad-Reza, S., *et al*. 2006. TCG Inside? A Note on TPM Specification Compliance. In Proceedings of the first ACM workshop on Scalable trusted computing. ACM: Alexandria, Virginia, USA.

[2] Ruhr-University. Chair for System Security-TPM Compliance Test. 2006 [cited 2009 October 18]; Available from: http://www.trust.rub.de/home/current-projects/tpmct/.

[3] Lee, D. and M. Yannakakis. 1996. Principles and Methods of Testing Finite State Machines-A Survey. Proceedings of the IEEE. 84(8): 1090–1123.

[4] Zhan, J., *et al*. 2008. Research on Automated Testing of the Trusted Platform Model. Zhang Jia Jie, Hunan, China: Inst. of Elec. and Elec. Eng. Computer Society.

[5] Zhang, H., *et al*. 2008. A Practical Solution to Trusted Computing Platform Testing. Wuhan, Hubei, China: Inst. of Elec. and Elec. Eng. Computer Society.

[6] Bourhfir, C., *et al*. 1997. Automatic Executable Test Case Generation for Extended Finite State Machine Protocols. In Testing of Communicating Systems. Springer. 75–90.

[7] Petrenko, A., S. Boroday, and R. Groz. 2004. Confirming Configurations in EFSM Testing. *Software Engineering, IEEE Transactions on*. 30(1): 29–42.

[8] Bochmann, G. V. and J. Gecsei. 1977. A Unified Method for the Specification and Verification of Protocols. Proceedings of IFIP Congress 77. 229–234.

[9] Li, H., H. Hu, and X.-F. Chen. 2009. Research on Compliant Testing Method of Trusted Cryptography Module. *Jisuanji Xuebao/Chinese Journal of Computers*. 32(4): 654–663.

[10] Bernhard, K., Oslo. 2007. Improving the Security of Trusted Computing, in Proceedings of 16th USENIX Security Symposium on USENIX Security Symposium. USENIX Association: Boston, MA.

[11] Kursawe, K., D. Schellekens, and B. Preneel. 2005. Analyzing trusted platform communication, in In: ECRYPT Workshop, CRASH – CRyptographic Advances in Secure Hardware. 8.

[12] Bruschi, D., *et al*. Replay attack in TCG specification and solution. 2005. Tucson, AZ, United states: IEEE Computer Society.

[13] Xu, S., *et al*. 2009. Security Analysis of OIAP Implementation based on BAN Logic. In 1st International Conference on Multimedia Information Networking and Security, MINES 2009. Hubei.

[14] Chen, L. and M. Ryan. 2009. Offline Dictionary Attack on TCG TPM Weak Authorisation Data, and Solution. In Future of Trust in Computing. 193-196.

[15] Gürgens, S., *et al*. 2008. Security Evaluation of Scenarios Based on the TCG's TPM Specification, in Computer Security–ESORICS 2007. 438–453.

[16] Delaune, S., *et al*. 2011. A Formal Analysis of Authentication in the TPM. In Formal Aspects of Security and Trust. Springer. 111–125.

[17] Fu, D., *et al*. 2013. Authentication of the Command TPM_CertifyKey in the Trusted Platform Module. *TELKOMNIKA Indonesian Journal of Electrical Engineering*. 11(2): 855–863.

[18] Tarnovsky, C. 2010. Deconstructing A `Secure' Processor. In Black Hat Brie_ngs Federal. http://www.blackhat.com/presentations/bh-dc-

10/Tarnovsky_Chris/BlackHat%-DC-2010-Tarnovsky-DASP-. February 2010.

[19] Jackson, W. Black Hat: Engineer Cracks 'Secure' TPM Chip. 2010; Available from: http://redmondmag.com/articles/2010/02/03/black-hat-engineer-cracks-tpm-chip.aspx.

[20] Chen, L., *et al*. 2009. Attacking the BitLocker Boot Process, in Trusted Computing. Springer Berlin / Heidelberg. 183–196.

[21] Wojtczuk, R. and J. Rutkowska. 2009. Attacking Intel Trusted Execution Technology in,In Black Hat DC, http://invisiblethingslab.com/resources/bh09dc/Attacking%20Intel%20TXT%20-%20paper.pdf. 2009.

[22] Li, H., D. Feng, and X. Chen. 2009. Compliant Testing Method of Trusted Cryptography Module [J]. *Journal of Wuhan University (Natural Science Edition)*. 1: 008.

[23] Xiao-Feng, C. 2009. The Formal Analysis and Testing of Trusted Platform Module. *Chinese Journal of Computers.* 32(4): 646–653.