DOI:10.16526/j.cnki.11-4762/tp.2011.11.073

计算机测量与控制. 2011. 19(11)
Computer Measurement & Control

文章编号:1671-4598(2011)11-2735-04　　中图分类号:TP393　　文献标识码:A

可信计算软件构架的检测研究

闫建红[1,2]　彭新光[1]

(1. 太原理工大学 计算机科学与技术学院，山西 太原　030024；
2. 太原师范学院 计算机系，山西 太原　030012)

摘要：介绍了 TSS 体系结构和 TSP 对象及其对象之间的关系，通过一个基于可信计算的数据密封程序，分析如何调用 TPM 驱动程序，如何使用驱动程序中各个对象，纵向说明了可信计算软件运行架构和过程；实验以 tpm _emulator0. 7. 1 模拟 TPM 芯片，验证了文件密封的过程和 TSS 的工作原理，展示了 TPM 芯片模拟、核心驱动程序、应用程序的三者之间的相互调用关系；对基于可信计算的软件编程起到很好的指导作用。

关键词：可信计算；TCG 软件栈；可信平台模块；TSP 对象

Study on Testing of TCG Software Stack

Yan Jianhong[1,2], Peng Xinguang[1]

(1. College of Computer and Software, Taiyuan University of Technology, Taiyuan　030024, China;
2. Department of Computer Science, Taiyuan Normal University, Taiyuan　030012, China)

Abstract: TSS (TCG Software Stack) architecture and relationship among class of TSS Service Providers (TSP) was introduced. Through one application program based on Trusted Computing, how to employ TCG Software Stack and the class of TSP was vertically analyzed. Tpm _emulator0. 7. 1 simulated TPM chip in experiment, the test verified the process of sealing file and working principle of TSS and showed the mutual relationship among application program terminal, TCS terminal and TPM terminal. The research has a great value in guiding the designing of trusted platform software application.

Key words: trusted computing; TCG software stack; trusted platform module; TSS service providers class

0　引言

可信计算的主要思路是在各种终端上引入可信架构。终端可信的核心是可信平台模块（Trusted Platform Module，TPM）的安全芯片[1-3]。可信计算组织（Trusted Computing Group，TCG）提出了 TCG 软件栈（TCG Software Stack，TSS)[4]。TSS 平台软件从结构上可以分为 3 层，自下至上分别为 TCG 设备驱动库 TDDL（TCG Device Driver Library)、TSS 核心服务 TCS（TSS Core Services）和 TCG 服务提供者 TSP（TSS Service Providers)，全部运行于用户模式。TSP 通过 TSPI（TSS Service Providers Interface）向上层应用程序提供接口[5]，在 TSS 规范中 TSP 涉及 7 个对象，分别是 Context、TPM、Policy、Key、EncData、PcrComposite、Hash 对象。可以将这 7 个对象看作 TPM 的 7 种不同功能。文献［6］给出了针对应用兼容问题的改进方案，但只是从局部命令入手，没有给出整体的兼容架构，文献［7］改进 TSS 以实现应用兼容，设计一种兼容的可信密码模块软件栈。文献［8］将可信计算理论引入其中，提出基于可信计算的 P2P 匿名通信系统。文献［9］对 TSP 对外提供的主要对象类进行分析并研究功能接口各个对象类之间的关系，文章只是理论上对象的关系进行论述，没有具体说明对象使用之间的相互关系和可信软件的层次结构。本文通过对数据密封的程序运行，分别从 3 个终端监测运行过程，说明 TCG 软件栈各个对象之间的调用关系。

收稿日期:2011-04-26；　修回日期:2011-06-06。

基金项目:本项目受山西省自然科学基金(2009011022-2)；山西省留学基金(2009-28)。

作者简介:闫建红(1972-)，女，山西孟县人，博士研究生，CCF 会员(E200015532G)，主要从事网络安全、可信计算方向的研究。

彭新光(1955-)，男，博士，博士生导师，主要从事计算机网络与安全方向的研究。

1　数据密封过程

密封过程是通过连接一个特定的 TPM，通过这个特定的 TPM 对其密封，最后仅仅能由这个特定的 TPM 去解封。通过对一个数据文件进行密封的应用程序的过程为例，分析如何使用 TPM 的软件栈，具体的 TSS 流程图如图 1 所示。

(1) 首先建立上下文，连接到本地的 tcs

①首先使用 Tspi _Context _Create (&hContext)，由 TSP 生成一个新的上下文对象，创建一个策略对象并与上下文关联，对于在这个上下文中创建的所有授权对象来说，这个策略对象是默认的策略。

②Tpsi _Context _Connect (hContext)，连接到本地的 TCS 提供者。

(2) 获取 TPM 对象，并产生一个随机数，这个随机数将来作为对称密钥被密封

①Tpsi _Context _GetTpmObject (hContext, &hTpm)，获取隐式创建的 TPM 的句柄，由于 TSS 会自动给 TPM 类对象分配一个 Policy 对象，因此程序没有给 TPM 对象获取策略。

②Tspi _TPM _GetRandom (hTpm, EVP _CIPHER _key _length (EVP _aes _256 _cbc ()), &randKey)，通过 TPM 产生随机数，这个随机数 randKey 将被作为对称密钥来

图 1 TSS 的应用程序调用流程

密封。

（3）**载入一个存储根密钥** SRK **密钥，并为之设置策略的秘密**

①Tspi _ Context _LoadkeybyUUID（hContext，TSS _PS _TYPE _SYSTEM，SRK _UUID，&hSrk），**载入新密钥的父密钥、存储根密钥，从系统永久存储区载入存储根密钥** SRK。

②Tpsi _GetPolicyObject（hSrk，&hSrkPolicy）**为密钥对象** SRK **获取策略对象。**

③Tspi _Policy _SetSecret（hSrkPolicy，（UINT32）pswd _len，（BYTE *）passwd）**设置策略对象的秘密。**

（4）**通过** SRK **产生并载入一个** RSA **密钥对象，并为之分配策略对象和设置秘密**

①Tpsi _Context _CreateObject（hContext，TSS _OBJECT _TYPE _RSAKEY，keyFlags，&hKey）**创建一个** RSA **密钥对象。**

②Tpsi _Context _CreateObject（hContext，TSS _OBJECT _TYPE _POLICY，TSS _POLICY _USAGE，&hPolicy）**为** RSA **密钥对象获取策略对象。**

③Tspi _Policy _SetSecret（hPolicy，strlen（TPMSEAL _SECRET），（BYTE *）TPMSEAL _SECRET）**为** RSA **密钥设置策略的秘密。**

④Tspi _Policy _AssigntOobject（hPolicy，hKey）**将策略对象分配给** RSA **密钥。**

⑤Tspi _Key _CreateKey（hKey，hSrk，NULL _HPCRS）**创建一个密钥。**

⑥Tspi _Key _LoadKey（hKey，hSrk）**载入这个密钥。**

（5）**创建一个数据加密对象并为之分配策略设置秘密**

①Tpsi _Context _CreateObject（hContext，TSS _OBJECT _TYPE _ENCDATA，TSS _ENCDATA _SEAL，&hEncdata）encdata）**创建一个关于加密数据对象。**

②Tpsi _Context _CreateObject（hContext，TSS _OBJECT _TYPE _POLICY，TSS _POLICY _USAGE，&hPolicy）**创建一个策略对象。**

③Tspi _Policy _SetSecret（hPolicy，strlen（TPMSEAL _SECRET），（BYTE *）TPMSEAL _SECRET）**设置该策略的秘密。**

④Tspi _Policy _AssignToObject（hPolicy，hEncdata）**将策略对象分配给加密数据对象。**

（6）**将随机数** randKey **作为对称密钥用** RSA **密钥加密，并将加密后数据放入** hEncdata **并通过获取属性函数得到数据。**

①Tspi _Data _Seal（hEncdata，hKey，EVP _CIPHER _key _length（EVP _aes _256 _cbc（）），randKey，hPcrs）**对数据进行密封，其中** PCR **是可选项。**

②Tspi _GetAttribData（hEncdata，TSS _TSPATTRIB _ENCDATA _BLOB，TSS _TSPATTRIB _ENCDATABLOB _BLOB，&encLen，&encKey）**获取加密对象的属性。**

整个过程首先建立一个 TPM **句柄，产生一个新的对称密钥** randKey，**使用该对称密钥对文件进行加密，并且使用** TPM **密钥密封了该对称密钥。整个过程应用程序是通过** TSPI **接口，调用相关的函数，再通过** TCS **使用** TDDL，**通过** TDDLI **接口使用** TPM。

2 实验测试

2.1 实验环境

本文测试硬件采用 Thinkpad R400 **笔记本电脑，使用** tpm _emulator0.7.1 **模拟** tpm **芯片，符合** TCG1.2 **规范，内存** 2GB，**内核为** 2.6.34.7，**操作系统** fedora；TSS **采用的是** Trousers 0.3.6。

2.2 应用程序终端测试

应用程序端是用来运行应用程序，按照上面的流程运行数据的密封过程如下，对其的 6 **个步骤进行详细的分析：**

（1）**调用** TSPI **接口，创建上下连接到本地的** TCS **接口**

TSPI rpc/tcstp/rpc.c：362 Sending TSP packet to host localhost.

TSPI rpc/tcstp/rpc.c：377 Connecting to 127.0.0.1

（2）**读取** PCR **信息，这个过程根据情况，在参数中如果没有** PCR，**将没有这一步，将产生一个随机数，作为对称密钥。**

TSPI rpc/tcstp/rpc _context.c：44 RPC _OpenContext _TP：Received TCS Context：0xa0082c02

rpc/tcstp/rpc _pcr _extend.c：69 RPC _PcrRead _TP：TCS Context：0xa0082c02

rpc/tcstp/rpc _pcr _extend.c：69 RPC _PcrRead _TP：TCS Context：0xa0082c02

TSPI rpc/tcstp/rpc _random.c：37 RPC _GetRandom _TP：TCS Context：0xa0082c02

（3）**载入本地** TPM **的** SRK **密钥，并设置密钥的策略**

TSPI rpc/tcstp/rpc _ps.c：318 RPC _LoadKeyByUUID _TP：TCS Context：0xa0082c02

TSPI rpc/tcstp/rpc _ps.c：339 RPC _LoadKeyByUUID _TP：TCS key handle：0x40000000

TSPI rpc/tcstp/rpc _ps.c：274 RPC _GetRegisteredKeyBlob _TP：TCS Context：0xa0082c02

Enter SRK password：

（4）**创建** RSA **密钥并创建策略，获取秘密并载入** RSA **密钥**

LOG __DEBUG TSPI rpc/tcstp/rpc __auth. c：70 RPC __OSAP __TP：TCS Context：0xa0082c02

LOG __DEBUG TSPI rpc/tcstp/rpc __key. c：119 RPC __CreateWrapKey __TP：TCS Context：0xa0082c02（创建 rsa 密钥）

LOG __DEBUG TSPI rpc/tcstp/rpc __auth. c：37 RPC __OIAP __TP：TCS Context：0xa0082c02（创建策略）

LOG __DEBUG TSPI obj __policy. c：230 Got a secret：（获取秘密）

DA 39 A3 EE 5E 6B 4B 0D 32 55 BF EF 95 60 18 90 AF D8 07 09

LOG __DEBUG TSPI rpc/tcstp/rpc __key. c：42 RPC __LoadKeyByBlob __TP：IN：TCS Context：0xa0082c02（载入密钥）

LOG __DEBUG TSPI rpc/tcstp/rpc __key. c：75 RPC __LoadKeyByBlob __TP：OUT：TCS key handle：0x22330001，TPM key slot：0x0

（5）创建加密对象

lOG __DEBUG TSPI rpc/tcstp/rpc __auth. c：70 RPC __OSAP __TP：TCS Context：0xa0082c02（创建数据加密对象）

（6）密封密钥，关闭上下文。

LOG __DEBUG TSPI rpc/tcstp/rpc __seal. c：46 common __Seal __TP：TCS Context：0xa0082c02

LOG __DEBUG TSPI rpc/tcstp/rpc __context. c：60 RPC __CloseContext __TP：TCS Context：0xa0082c02

从上面的运行情况分析，整个过程的上下文句柄编号为 0xa0082c02，运行这个应用程序，经过测试，加密一个容量为 257k 的文件，加密之后的容量为 360k；一个容量为 1121k 的文件，加密之后大小为 1567k，都比原文件的容量要略大一些。通过远程过程调用服务 RPC（Remote Procedure Call Server）使用 TCS，负责可信平台之间的数据传递和功能调用。

2.3　TCSD 终端

TCSD 是 TPM 的核心服务，对上连接应用程序，对下连接 TDDL，TCS 接口实现的类 C 函数可以解析为 TPM 需要的参数块，TCS 对资源提供本地或远程的调用方式。

由于运行过程繁琐，本文仅对其中的获取随机数过程进行分析：

TCSD tcsd_threads. c:381 Rx'd packetTCSD TCS rpc/tcstp/rpc. c:577 Dispatching ordinal 44

TCSD TCS rpc/tcstp/rpc __random. c:41 tcs __wrap __GetRandom: thread −1217762416 context a092d100

TCSD TCS tcsi __random. c: 48 TCSP __GetRandom __Internal: 32 bytes

To TPM: 00 C1 00 00 00 0E 00 00 00 46 00 00 00 20

TCSD TDDL tddl. c:171 Calling write to driver

From TPM: 00 C4 00 00 00 2E 00 00 00 00 00 00 00 20 DF F8

From TPM: F5 6F 60 88 C5 3D 37 70 78 FF 6C D1 4A 79 D3 C1

From TPM: 26 97 4D 7A A5 7F C0 B1 87 C0 C1 0A 5A 9A

TCSD tcsd_threads. c:408 Sending 0x42 bytes back

从这段运行过程可以看出，它连接驱动程序 trousers 的 rpc/tcstp/rpc. c 等，通过调用 tddl. c 对 TPM 进行读写，TDDL 是 TSS 设备驱动程序的接口。经过分析这个随机数的获取过程，从核心服务写数据到 TPM，以及从 TPM 读数据到核心服务，整个过程和应用程序端的过程是一致的。

2.4　tpmd 终端

这个终端主要是监测 tpm __emularor 中 TPM 的命令运行，通过应用程序调用驱动程序 TSPI 程序，再调用 TCS，通过 TDDL 调用 TPM 命令。这里省略详细的调试过程，只是显示使用到的主要 TPM 命令。

（1）获取 tpm 随机数作为对称密钥

tpm_cmd_handler. c:3749: Debug: [TPM_ORD_GetRandom]

tpm_crypto. c:218: Info: TPM_GetRandom()

（2）获得 RSK 密钥并载入 rsa 密钥

tpm_authorization. c:175: Info: TPM_OSAP()

tpm_storage. c:636: Info: TPM_CreateWrapKey()

tpm_authorization. c:526: Info: tpm_verify_auth()

tpm_authorization. c:156: Info: TPM_OIAP()

tpm_capability. c:697: Info: TPM_GetCapability()

tpm_storage. c:844: Info: TPM_LoadKey2()

tpm_storage. c:744: Info: TPM_LoadKey()

tpm_authorization. c:526: Info: tpm_verify_auth()

tpm_eviction. c:51: Info: TPM_FlushSpecific()

（3）用 RSA 密钥加密对称密钥，PCR 作为可选项进行加密

tpm_authorization. c:175: Info: TPM_OSAP()

tpm_capability. c:697: Info: TPM_GetCapability()

tpm_storage. c:362: Info: TPM_Seal()

(tpm_integrity. c:139: Info: tpm_compute_pcr_digest())

tpm_eviction. c:51: Info: TPM_FlushSpecific()

在 tpmd 端，由于无需 TSP 对象的创建，只是与 TPM 相关的系统命令。因此这一端不会显示应用程序端的部分对应的过程。从运行过程，可以看出，这部分和 TCS 的运行也是一致的。

3　实验分析

本实验通过 3 个终端进行监测：一个终端用来监测 tpm __emulator 运行，即 tpmd 终端；一个终端用来监测核心服务层监测 tcsd 的运行；一个终端用来运行应用程序。从硬件、核心驱动程序和应用程序 3 个方面纵向说明其工作流程以及如何使用 TSP 进行编程。在对象的相互关系中，必须首先申请上下文对象并创建 ContextObject，创建策略对象 PolicyObject，并对策略进行设置后与上下文相关联。在实验中，应用程序通过核心服务层的接口，使用 TPM 模拟器来执行，实现了基于可信计算的数据加密。整个过程使用 TCS 接口，读取 TPM 的 PCR 的值，使用 TPM 的 SRK 密钥创建 RSA 密钥，对加密对象进行加密、密封密钥。在应用程序端关闭上下文。在驱动程序端通过 TCG 设备驱动库 TDDL 对 TPM 进行读写，而在 tpmd 终端，使用 TPM 相关命令对 TPM 进行操作完成加密过程，通过获取随机数，使用特定对象授权协议（OSAP）和对象无关授权协议（OIAP），由 TPM 创建并存储密钥并向 TPM 装载密钥，并密封数据将摘要存储到 PCR 中。通过监测 3 个终端的运行过程，展示了 TPM 芯片模拟、核心驱动程序、应用程序的三者之间的相互调用关系，以及如何使用应用程序对核心驱动程序和 TPM 模拟器的调用。例如：在程序中使用 Tspi __TPM __GetRandom 接口通过应用程序端的检测，在 TSPI 接口中运行 TSPI rpc/tcstp/rpc __random. c：37 RPC __GetRandom __TP，在 TCSD 端使用 tcsi __random. c：48 TCSP

_GetRandom _Internal，在 tpmd 端通过 tpm _crypto. c：218：Info：TPM _GetRandom () 来使用 TPM 产生随机数。

4 结束语

可信平台模块 TPM 是可信计算技术的关键组成部件，对其进行软件架构的测试不但能方便用户使用，还能为其安全性提供分析基础，降低安全风险。本文通过一个基于可信计算的数据密封的程序，从应用程序端、TCS（TSS Core Services）端、TPM 端 3 个方面，不同的视觉，纵向深入说明了可信计算运行的过程，即通过应用程序调用 TCSD 的接口，TCS 通过 TDDL 调用 TPM 的过程。在实验的基础上，深入分析如何使用 TPM 驱动程序，使用驱动程序的各个对象，以及在调用过程中，必须要申请上下文对象，创建上下文相关的对象，创建策略对象并设置，最后与对象相连等过程。验证了 TSS 的工作原理，更好地指导基于 TSS 编程和可信应用软件的编写。

参考文献：

[1] Trusted Computing Group. TPM Main Part 1 Design Principles [EB/OL]. http://www.trustedcomputinggroup.org/In Jan. 2011.

[2] Trusted Computing Group. TPM Main Part 2 TPM Structures [EB/OL]. http://www.trustedcomputinggroup.org/In Jan. 2011.

[3] Trusted Computing Group. TPM Main Part 3 Commands [EB/OL]. http://www.trustedcomputinggroup.org/In Jan. 2011.

[4] Trusted Computing Group. TCG Software Stack Specification Version 1.2 [EB/OL]. http://www.trustedcomputinggroup.org/In Jan. 2011.

[5] Challener D.，等. 可信计算 [M]. 赵　波，等译北京：机械工业出版社，2009.

[6] 刘　毅，沈昌祥. 一种可信软件栈的兼容性改进方案 [J]. 武汉大学学报（理学版），2009，55 (1)：57-61.

[7] 张　兴，黄宁玉，祝　璐. 可信密码模块软件栈兼容方案设计 [J]. 武汉大学学报（武汉大学学报），2010，35 (5)：618-621.

[8] 任　帅，慕德俊. 基于可信计算的 P2P 匿名通信系统 [J]. 计算机测量与控制，2009，17 (5)：965-976.

[9] 王震宇，窦增杰，田　佳，等. TCG 软件栈功能模型分析 [J]. 信息工程大学学报，2010，(2)：71-74.

（上接 2734 页）

仿真条件：高度 8000m，$Ma=1.8$，期望攻角指令 α_d（幅值为 40°，周期为 4s），期望侧滑角为 β_d（幅值为 5°，周期为 4s），期望滚转角为 γ_d（幅值为 0°，周期为 4s）。仿真结果如图 1～3 所示。仿真结果表明，攻角、侧滑角及滚转角均能够很好地响应系统指令，且控制系统动态品质良好，解耦效果明显，系统具有良好的鲁棒性，达到了设计目的。

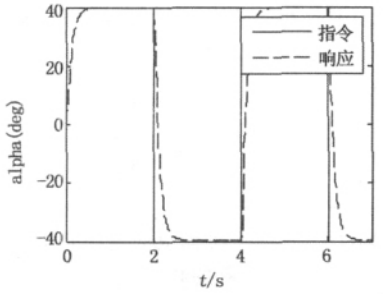

图 1　攻角响应曲线

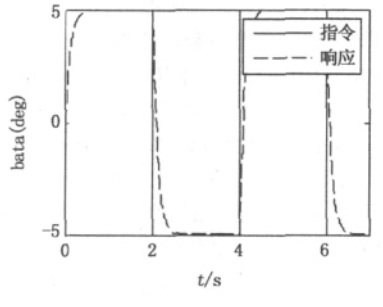

图 2　侧滑角响应曲线

4 结束语

本文在大攻角空空导弹非线性数学模型的基础上，采用基于神经网络动态逆方法进行了导弹的解耦控制设计。仿真结果表明了设计方案的可行性，解耦控制系统具有良好的动态品质，所作工作可对大攻角导弹解耦工程实现具有一定的参考作用。

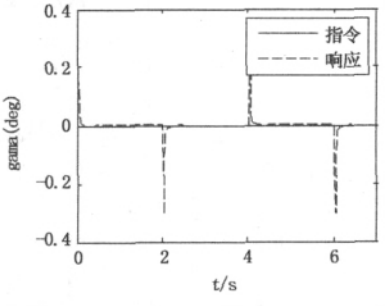

图 3　滚转角响应曲线

参考文献：

[1] 罗绪涛，等. 基于 PV 规范型的大攻角导弹解耦控制器设计 [A]. 中国科协第十届年会论文 [C]. 2008.

[2] 赵　阳，等. 基于神经网络和遗传算法的悬挂系统优化设计 [J]. 计算机测量与控制，2005，13 (10)：1083-1084.

[3] 张友安，胡云安，苏身榜. BTT 导弹控制系统鲁棒动态逆设计 [J]. 宇航学报，2002，23 (2)：89-91.

[4] Lee H P，Reiman S E，et al. Robust nonlinear dynamic inversion control for a hypersonic cruise vehicle [A]. AIAA Guidance，Navigation and Control Conference and Exhibit [C]. Hilton Head，South Carolina，2007.

[5] McFarland M B，Hoque S M. Robustness of a nonlinear missile autopilot designed using dynamic inversion [A]. AIAA Guidance，Navigation，and Control Conference and Exhibit [C]. Denver，CO. 2000.

[6] Haga R，Matsuura A，et al. Neural Network Based Adaptive Flight Control Using Feedback Error Learning [A]. AIAA Guidance，Navigation，and Control Conference and Exhibit [C]. 21-24 August 2006，Keystone，Colorado.