一种 TPM2.0 密钥迁移协议及安全分析

宋 敏[1],谭 良[1,2]
(1. 四川师范大学计算机科学学院,四川成都 610101;2. 中国科学院计算技术研究所,北京 100190)

摘 要: 国际规范《TPM-Rev-2.0-Part-1-Architecture-01.38》允许用户基于密钥复制接口来设计密钥迁移协议以实现芯片间密钥的共享,并在复制过程中通过 innerwrap 和 outerwrap 为复制密钥提供机密性、完整性和认证性. 然而通过分析发现基于密钥复制接口设计的密钥迁移协议存在三个问题:(1)缺少交互双方 TPM 的相互认证,会导致密钥能够在敌手和 TPM 间迁移;(2)当复制密钥的属性 encryptedDuplication = 0 且新父密钥的句柄 newParentHandle = TPM_RH_ NULL 时,复制接口不能实施 innerwrap 和 outerwrap,复制密钥将以明文传输而造成泄露;(3)当新父密钥是对称密钥时,innerwrap 中的对称加密密钥以及 outerwrap 中的密钥种子如何在源 TPM 与目标 TPM 之间安全交换,《TPM-Rev-2.0-Part-1-Architecture-01.38》并没有给出具体的解决办法. 不仅如此,该协议流程还非常复杂,也无法统一,容易导致复制密钥泄露. 针对以上问题,本文提出了基于 MAK(Migrate Authentication Key)和复制接口的新密钥迁移协议—MDKMP(Make Duplication Key Migration Protocol). 首先在 TPM 中的 SRK 下新增只用于 TPM 密钥迁移的非对称密钥 MAK,并申请相应证书—TPM 迁移认证证书;然后基于 MAK 证书和复制接口设计新的 TPM 密钥迁移协议 MDKMP. MDKMP 采用两阶段迁移模式,第一阶段将源 TPM 密钥迁移到目的 TPM 的 MAK 下,第二阶段再将密钥迁移到新父密钥下.

关键词: 可信计算; TPM2.0; 迁移证书; 密钥复制; 密钥迁移
中图分类号: TP309 **文献标识码:** A **文章编号:** 0372-2112 (2019)07-1449-16
电子学报 URL: http://www.ejournal.org.cn **DOI:** 10.3969/j.issn.0372-2112.2019.07.008

A TPM2.0 Key Migration Protocol and Security Analysis

SONG Min[1], TAN Liang[1,2]
(1. *College of Computer Science, Sichuan Normal University, Chengdu, Sichuan* 610101, *China*;
2. *Institute of Computing Technology, Chinese Academy of Sciences, Beijing* 100190, *China*)

Abstract: TPM specification allows users to design key migration protocol to share keys, and the key is provided with confidentiality, integrity and authentication through innerwrap and outerwrap process. However, it is found that there are three problems in this protocol: (1) lack of authentication, which results in the fact that the key can be migrated between adversary and TPM; (2) when encryptedDuplication = 0 and newparentHandle = TPM_RH_NULL, it cannot implement innerwrap and outerwrap, the key will be leakaged; (3) when the new parent key is a symmetric key, how will the symmetric encryption key and the seed exchange securely between the source TPM and the target TPM. To solve the above problems, a new key migration protocol MDMKP is proposed. The MDMKP uses a two-phase migration mode. In the first phase, the key is migrated to the MAK of the destination TPM. In the second phase, the key is migrated to the new parent key.

Key words: trusted computing; TPM2.0; migration certificate; key duplication; key migration

1 引言

可信计算技术的基本思想是:在通用计算平台上嵌入一个防篡改的硬件可信安全芯片,利用芯片的安全特性保证系统按照预期的行为执行,从根本上提高终端的安全性[1,2]. TPM (Trusted Platform Module,

收稿日期:2018-06-11;修回日期:2018-11-12;责任编辑:孙瑶
基金项目:国家自然科学基金(No.61373162);四川省科技支撑项目(No.2014GZ0007);可视化计算与虚拟现实四川省重点实验室项目(No.KJ201402)

TPM)[2]是国际广泛使用的,是符合可信平台模块标准的安全芯片.具有密码学功能和受保护的存储空间,能够为可信计算平台提供密钥管理、平台数据保护、完整性存储与报告、身份标识等功能[3~8].

TPM 密钥管理是其非常重要的功能,它是 TPM 能够有效地提供其他各项功能的前提和基础.为了满足密钥的安全存储、分发和备份,TPM 采用层次型的存储保护体系,并提供密钥迁移(复制)接口.

TPM1.1[5,6]规范中定义的密钥迁移接口共三个,分别是

TPM_AuthorizeMigrationKey()

TPM_CreateMigratedBlob()

TPM_convertMigratedBlob()

TPM1.2[9~11]规范定义的密钥迁移接口共六个,分别是

TPM_AuthorizeMigratinKey()

TPM_CMK_ApproveMA()

TPM_CMK_CreateKey()

TPM_CMK_CreateTicket()

TPM_CMK_CreateBlob()

TPM_CMK_ConvertMigration()

TPM2.0[12]规范定义的密钥复制接口共两个,分别是

TPM2_Duplicate()

TPM2_Import()

通常,用户或上层应用可以通过以上接口设计密钥迁移协议,将源 TPM 中的密钥迁移到目的 TPM 中,以实现 TPM 芯片间密钥的共享.为了保证整个迁移密钥的安全,需要在整个过程保证迁移密钥的机密性、完整性和认证性.

然而,TPM2.0 的密钥迁移协议设计会更加复杂.一方面,由于 TPM2.0 已支持对称密钥对象,使得在设计密钥迁移协议时需要考虑更多的迁移需求组合.因为迁移密钥既可以是对称密钥也可以是非对称密钥,而新父密钥也既可以是对称密钥也可以是非对称密钥,不同的迁移需求组合在协议设计过程中需要进行不同的考虑;另一方面,TPM2.0 是通过密钥对象的复制属性(fixedTPM,fixedParent,encryptedDuplication)和新父密钥的句柄类型(newParentHandle)来决定迁移方式和迁移过程.不同的密钥属性和不同的密钥类型有不同的迁移组合,其迁移方式和迁移过程也就不同,特别是在密钥复制过程中是否进行 innerwrap 和 outerwrap,是保证密钥迁移机密性、完整性的关键.通过进一步研究发现,基于密钥复制接口的密钥迁移协议至少存在三个问题:其一是缺少交互双方 TPM 的相互认证,会导致密钥能够在敌手和 TPM 间迁移;其二是当迁移密钥的属性 encryptedDuplication = 0 且新父密钥的句柄 newParentHandle = TPM_RH_NULL 时,复制接口不能实施 innerwrap 和 outerwrap,迁移密钥将以明文传输而造成泄露;其三是当新父密钥是对称密钥时,innerwrap 中的对称加密密钥以及 outerwrap 中的密钥种子如何在源 TPM 与目标 TPM 之间安全交换,《TPM-Rev-2.0-Part-1-Architecture-01.38》并没有给出具体的解决办法.

针对以上问题,本文首先在 TPM 中的 SRK 下新增只用于 TPM 密钥迁移的非对称密钥 MAK(Migrate Authentication Key),并申请相应证书——TPM 迁移认证证书;然后基于 MAK 证书和复制接口提出了新的 TPM 密钥迁移协议.该协议采用两阶段迁移模式,第一阶段将源 TPM 密钥迁移到目的 TPM 的 MAK 下,第二阶段再将密钥迁移到新父密钥下.安全分析表明,该协议不仅能很好地解决前面那三个问题,而且能够统一协议流程,降低复杂度,保证复制密钥不会泄露.最后,通过模拟实验验证显示,该协议满足正确性和预期的安全属性.

2 背景知识

本节从 TPM 2.0 的密钥类型和结构、密钥对象管理保护体系、密钥迁移类接口及 innerwrap 和 outerwrap 过程等 4 个方面介绍本文的背景知识.

2.1 TPM2.0 的密钥类型和结构

相比于 TPM1.2,TPM2.0 的密钥类型和结构发生了较大的变化,下面我们对此进行介绍.

2.1.1 TPM2.0 密钥类型

(1)按照密钥功能组合分类

TPM2.0 中密钥对象的基本属性包括:

RestrictedAttribute 专用属性,表明该密钥只能对特定对象进行操作;

Sign Attribute 签名属性,表明该密钥对象是否可以用于签名;

Decrypt Attribute 机密属性,表明该对象是否可以用于加解密.

根据密钥对象以上三个属性组合,TPM2.0 将密钥对象分为 8 类,具体参见国际标准《TPM-Rev-2.0-Part-1-Architecture-01.38》.

(2)按照密钥复制属性分类

TPM2.0 中密钥对象除了基本属性外,还包括一些其他属性,如 fixedTPM 和 fixedParent、stclear、sensitiveDataOrigin、userWithAuth、adminWithPolicy、noDA 和 encryptedDuplication.根据 fixedTPM 和 fixedParent 的属性组合,可以将密钥对象分为可复制密钥和不可复制密钥.如表 1 所示.本文所指的可复制密钥,就是指 fixedT-

PM = 0 和 fixedParent = 0 的密钥.

表 1　密钥可复制分类表

fixedTPM	fixedParent	描述
0	0	密钥可以被复制
0	1	密钥跟随父密钥复制
1	0	无此类组合
1	1	不可复制密钥

2.1.2　TPM2.0 密钥结构

TPM2.0 中密钥对象的基本结构包括三个域:PublicArea、SensitiveArea 或 PrivateArea,对 TPM2.0 内部的任意密钥 k,表示为 k = (PublicArea, SensitiveArea),外部的任意密钥,通常表示为 k = (PublicArea, PrivateArea),其定义如下所示.

(1) Public Area

(a) type:密钥类型;

(b) nameAlg:此密钥支持的密码算法;

(c) objectAttributes:密钥属性,包括功能(Sign、Decrypt 和 Restricted)、授权(userWithAuth、adminWithPolicy 和 noDA)、复制(fixedTPM、fixedParent 和 encryptedDuplication)、生成方式(sensitiveDataOrigin)以及重置(stclear);

(d) authPolicy:授权策略;

(e) parameters:此类密码算法的参数;

(f) unique:非对称密钥此项代表公钥;对称密钥此项代表其 SensitiveArea 的摘要值.

(2) Sensitive Area

(a) sensitiveType:敏感数据类型;

(b) authValue:授权值;

(c) seedValue:对于对称的和非对称的存储密钥,由该值产生保护 child 对象的密钥. 对于非对称密钥的非存储密钥当前无用. 对于其他对象,该值与 sensitive 一起 HASH 产生 unique 摘要值;

(d) sensitive:敏感参数值. 对于非对称密钥,此值表示私钥,对于对称密钥,此值就是 key,对于消息码,此值就是 key;对于数据对象,此值就是敏感数据.

(3) Private Area

(a) encypted sensitive area:此密文是由父密钥的 seedValue 产生的 key 对 sensitive area 加密的值;

(b) HMAC 消息码:此消息码是由父密钥的 seedValue 产生的 key 对 sensitive area 进行 HMAC 的值.

在创建不同类型的对象时,可以调用三个接口 TPM2_CreatePrimary()、TPM2_Create()和 TPM2_CreateLoaded(). 创建的对象类型依赖于输入参数 ParentHandle 的类型. 函数执行成功会返回 PublicArea 和 SensitiveArea,前两个接口还会返回 TPMS_CREATION_DATA 类的 creationData.

2.2　TPM2.0 密钥对象管理存储保护体系

TPM2.0 可通 TPM2_CreatePrimary、TPM2_Create 和 TPM2_CreateLoaded 生成种子密钥对象、普通密钥对象和派生密钥对象,所有的密钥对象形成一颗密钥树,其中种子密钥对象一般作为根密钥保护所在层次的子密钥,如图 1 所示. 在此密钥树中,TPM2.0 将密钥对象分为可复制密钥对象、可跟随父密钥复制密钥对象以及不可复制密钥对象. 不可复制密钥对象只能与原 TPM 芯片绑定,不能被复制;而可复制密钥对象可以复制到其他 TPM 中使用,而可跟随父密钥复制密钥对象的存在使得在密钥树中一次能复制一颗子树,比 TPM1.2 中的密钥迁移更灵活,效率更高.

在 TPM2.0 密钥树中,存储父密钥采用对称加密方法保护孩子密钥,加密密钥由父密钥的密钥种子 seedValue 产生,加密算法由父密钥的 nameAlg 指定. 这一点与 TPM1.2 中用非对称密钥的私钥对孩子的私钥进行加密保护不同. 因此,在 TPM2.0 的密钥树中,无论是对称密钥还是非对称密钥,只要是存储密钥,都可以作为父节点. 如图 2 所示.

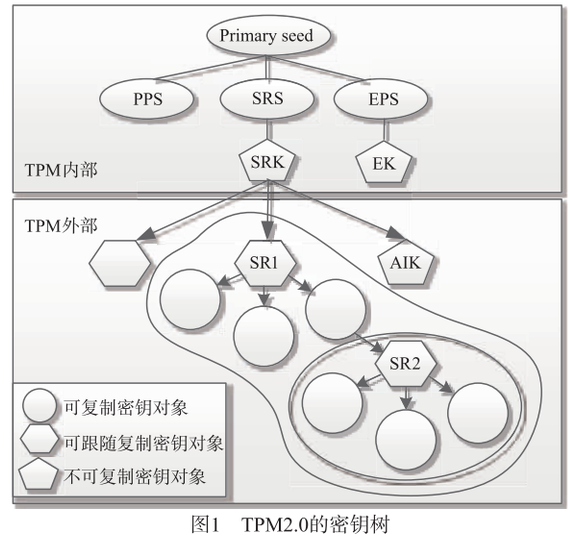


图1　TPM2.0的密钥树

2.3　TPM2.0 密钥对象复制接口

TPM2.0 完整的密钥复制流程是将源 TPM 的密钥对象复制到目标 TPM. 因此,密钥复制应该包括两个接口,其一是复制数据的生成;其二是复制数据的加载.

接口 1(复制数据生成接口)

TPM2_Duplication (objectHandle, newParentHandle, encrptionKeyIn, symmetricAlg). 其中 objectHandle 为复制密钥的句柄,newParentHandle 是新父密钥句柄,encryptionKeyIn 是 innerwrap 加密密钥,该密钥或由 caller 传入,或是由 TPM 产生,symmetricAlg 是对称加密算法. 该

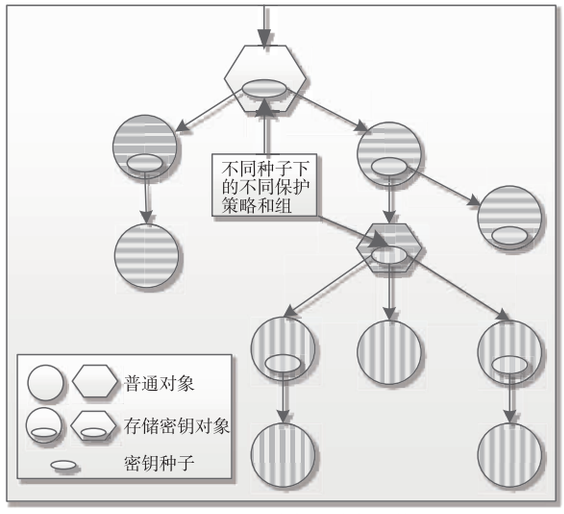


图2 TPM2.0的密钥层次保护模型

函数执行后返回三个值,其一是 encryptionKeyOut,encryptionKeyOut 返回的是由 TPM 产生的内部加密密钥,如果 TPM 没产生内部加密密钥,该值返回 null;其二是 duplicate,duplicate 是复制数据,封装了被复制密钥的 SensitiveArea;最后是 outSymSeed,outSymSeed 是 outerwrap 密钥种子,由它可以产生外部对称加密的密钥.

该接口的执行过程如下:

(1)检查迁移密钥的属性 fixedTPM 和 fixedParent,如果设置不是(0,0)就结束复制过程;

(2)检查迁移密钥的属性 encryptedDuplication,如果设置为 1,则判断 newParentHandle 是否为 TPM_RH_NULL,如果为 TPM_RH_NULL,则结束复制过程;否则转到(3);如果设置为 0 且 newParentHandle 不为 TPM_RH_NULL,则转到(4);

(3)执行 innerwrap,用 encrptionKeyIn 对复制密钥的 sensitiveArea 进行加密,生成 encSensitive;

(4)执行 outerwrap,用密钥种子 seed 生成加密密钥和 HMAC 密钥,对 encSensitive 进行加密和 HAMC 运算,得到 dupSensitive 和 outerHMAC.

接口 2(复制数据导入接口)

TPM2_Import(newparentHandle,encryptionKey,duplicate,inSymSeed). 其中 newparentHandle 是新父密钥句柄,encryptionKey 是源 TPM 内的 innerwrap 密钥,其值由 TPM2_Duplicate 的返回值 encryptionKeyOut 提供,duplicate 是复制数据,此值由 TPM2_Duplicate 的返回值 duplicate 提供,inSymSeed 是源 TPM 内的 outerwrap 密钥种子,此值由 PM2_Duplicate 的返回值 outSymSeed 提供.

该接口的执行过程如下:

(1)检查复制密钥的属性 fixedTPM 和 fixedParent,如果设置不是(0,0)就结束导入过程;

(2)检查新父密钥是否为存储密钥,如不是,结束导入过程;

(3)检查 innerwrap 的加密密钥 encryptionKey 是否正确,如果不正确,结束导入过程;

(4)检查 outerwrap 的密钥种子 inSymSeed 是否正确,如果不正确,结束导入过程;

(5)由 inSymSeed 和 newparentHandle 恢复出源 TPM 的 outerwrap 的 HAMC 密钥,通过 outerHAMC 验证 dupsensitive 及其 name 的真实性和完整性;然后再恢复出 outerwrap 的加密密钥,对 dupsensitive 解密得到 encsensitive;

(6)用 encryptionKey 对 encsensitive 进行解密,得到被复制密钥的 sensitive;并对 sensitive 及其 name 进行完整性校验.

通过对 TPM2_Duplication 接口和 TPM2_Import 接口的分析可知,采用 TPM2.0 的密钥复制接口来设计密钥迁移协议,需要将新父密钥从目标 TPM 传递到源 TPM,源 TPM 调用 TPM2_Duplication 接口得到被迁移密钥的复制数据,目标 TPM 调用 TPM2_Import 将复制数据载入. 基本的流程如下:

(1)目标 TPM 将新父密钥传递给源 TPM;

(2)源 TPM 调用 TPM2_Duplication 接口,根据迁移密钥的复制属性(fixedTPM,fixedParent,encryptedDuplication)和新父密钥的 newParentHandle 类型实施 innerwrap 和 outerwrap,得到复制数据;

(3)将复制数据传递给目标 TPM,目标 TPM 调用 TPM2_Import 接口将被迁移密钥加载到新父密钥下.

2.4 innerwrap 和 outerwrap 过程

由 2.3 节可以看出,在利用 TPM2_Duplication 接口和 TPM2_Import 接口进行密钥迁移时,复制过程的安全性不仅依赖 innnerwrap,而且依赖 outerwrap,下面我们对 innnerwrap 和 outerwrap 进行分析.

2.4.1 innerwrap 过程

通过对《TPM-Rev-2.0-Part-1-Architecture-01.38》和《TPM-Rev-2.0-Part-3-Commands-01.38-code》分析发现,密钥复制过程中是否进行 innerwrap 是由迁移密钥的属性 encryptedDuplication 和新父密钥的密钥句柄类型决定,只有当 encryptedDuplication = 1 且 newParentHandle! = TPM_RH_ NULL 时,innerwrap 才能发生. innerwrap 过程中需要的对称加密密钥由 caller 决定,caller 可以选择输入,也可以选择由 TPM 自行产生.

innerwrap 可以在复制过程中为迁移密钥提供完整性和机密性,包括两个步骤,首先是用复制密钥的 Hash 算法计算复制密钥 sensitive 和 name 的哈希值 innerIntegrity,保证复制密钥的完整性. 然后用某对称加密算法

的 CBF 模式对 innerIntegrity ‖ sensitive 进行加密得到 encSensitive. 其中需要的对称加密算法和密钥由 caller 将其作为参数传入 TPM2_Duplication.

从以上过程可以看出,源 TPM 要完成 innerwrap,需要确保加密密钥安全的传递到目的 TPM. 因此,如何将此加密密钥安全地传递到目的 TPM 是需要考虑的重要问题. 需要考虑如下 3 种情况:

(1)如果新父密钥是非对称密钥,无论迁移密钥是对称密钥还是非对称密钥,innerwrap 中需要的对称加密密钥可以由源 TPM 产生或由 caller 输入,且可以采用《TPM-Rev-2.0-Part-1-Architecture-01.38》中 B.10.3 或 C.6.3 中规定的算法进行保护交换.

(2)如果新父密钥是对称密钥,而复制密钥是非对称密钥,则 innerwrap 中需要的对称加密密钥可以由目的 TPM 产生,且可以采用《TPM-Rev-2.0-Part-1-Architec-ture-01.38》中 B.10.3 或 C.6.3 中规定的算法进行保护交换.

(3)如果新父密钥是对称密钥,而迁移密钥也是对称密钥,则 innerwrap 中需要的密钥无论是由源 TPM 产生或目的 TPM 产生或 caller 输入,《TPM-Rev-2.0-Part-1-Architecture-01.38》中都未给出此密钥的保护交换办法.

2.4.2　outerwrap 过程

通过对《TPM-Rev-2.0-Part-1-Architecture-01.38》和《TPM-Rev-2.0-Part-3-Commands-01.38-code》分析发现,密钥复制过程中是否进行 outerwrap 是由新父密钥的密钥句柄类型决定,当 newParentHandle! = TPM_RH_NULL 时,outerwrap 就会发生,当 newParentHandle = TPM_RH_NULL 时,outerwrap 不会发生. 究其原因是 outerwrap 过程主要由新父密钥控制,如果新父密钥的句柄为 TPM_RH_NULL,则新父密钥不能为 outerwrap 提供需要的算法及参数.

outerwrap 可以在复制过程中为迁移密钥提供机密性、完整性以及对新父密钥的认证性. 包括两个步骤,第一是对 encSensitive 进行加密. 具体过程为:首先获得密钥种子 seed,然后将此密钥种子、新父密钥的 npNameAlg 和迁移密钥的 Name 作为 KDFa()的参数产生对称加密密钥,最后采用新父密钥的对称加密算法对 encSensitive 进行加密得到 dupSensitive;第二是对 dupSensitive 和 Name 进行 HAMC 运算,产生消息码 outerHMAC. 具体过程为:首先用 Seed 和新父密钥的 npNameAlg 作为 KDFa()的参数产生 HAMC 密钥,然后采用新父密钥的 HMAC 算法进行运算获得消息码 outerHMAC.

从以上过程可以看出,源 TPM 要完成 outerwrap,不仅需要获得新父密钥的相关参数,而且需要确保密钥种子 seed 能在源 TPM 和目的 TPM 之间安全地交换. 需要考虑的三种情况与 innerwrap 要考虑的三种情况一致.

3　复制接口的密钥迁移协议及问题分析

通过对 TPM2_Duplication 接口和 TPM2_Import 接口的分析可知,采用 TPM2.0 的密钥复制接口来设计密钥迁移协议,需要将新父密钥从目标 TPM 传递到源 TPM,源 TPM 调用 TPM2_Duplication 接口得到被迁移密钥的复制数据,目标 TPM 调用 TPM2_Import 将复制数据载入. 基本的流程如下:

(1)目标 TPM 将新父密钥传递给源 TPM;

(2)源 TPM 调用 TPM2_Duplication 接口,根据迁移密钥的复制属性(fixedTPM,fixedParent,encryptedDuplication)和新父密钥的 newParentHandle 类型实施 innerwrap 和 outerwrap,得到复制数据;

(3)将复制数据传递给目标 TPM,目标 TPM 调用 TPM2_Import 将被迁移密钥加载到新父密钥下.

我们将在 3.1 节给出详细的密钥迁移协议流程,3.2 节对该协议进行分析.

3.1　基于复制接口的密钥迁移协议

基于密钥复制接口来设计密钥迁移协议通常包括 6 个参与实体:源 TPM、源 TPM 的所有者、目标 TPM、目标 TPM 的所有者、源 TPM 所在的主机、目标 TPM 所在的主机,其中,前 4 个实体是可信的,而源和目标 TPM 所在的主机被认为是不可信的. 表 2 为协议相关符号及函数说明.

表 2　协议相关符号及函数说明

符号/函数	描述
T_S,O_S,H_S	源 TPM、源 TPM 的所有者、源 TPM 所在的主机
T_D,O_D,H_D	目标 TPM、目标 TPM 的所有者、目标 TPM 所在的主机
newParent,newParentHandle	新父密钥,新父密钥句柄
object,objectHandle	复制密钥,复制密钥句柄
encryptionKeyin	innerwrap 的对称密钥
Seed	outerwrap 的密钥种子
RSA_OAEP,ECC_ECDH	RAS 和 ECC 加密算法
symmetricAlg	源 TPM 和目标 TPM 都支持的对称加密算法

根据第 2 节的分析,并基于《TPM-Rev-2.0-Part-1-Architecture-01.38》规范的要求,我们设计出基于复制接口的密钥迁移协议,其流程如下.

(1)$H_D \rightarrow H_S$:newParent. publicAera,newparentHandle,

symmetricAlg.

(2) O_S:

(a)获得 objectHandle,newparentHandle;

(b)决定 encryptionKeyin 的产生方式,可以输入,可以由 TPM 内部产生 RNG;

(c)调用接口 TPM2_Duplication (objectHandle,newParentHandle,encrptionKeyIn,symmetricAlg).

(3) T_S: 执行复制过程.

(a)检查复制密钥的属性 fixedTPM 和 fixedParent,如果设置不是(0,0)就结束复制过程,转到(10);

(b)检查迁移密钥的属性 encryptedDuplication,如果设置为 1,则判断 newParentHandle 是否为 TPM_RH_NULL,如果为 TPM_RH_NULL,则结束复制过程,转到(10);否则转到第(3)步(c);如果设置为 0,且 newParentHandle 不为 TPM_RH_NULL 则转到第(3)步(e),否则转向第(3)步(g);

(c)执行 innerwrap,由 caller 输入或 TPM 产生 encrptionKeyIn,并用 encrptionKeyIn 对复制密钥的 sensitiveArea 进行加密,生成 encSensitive;

(d)对 encrptionKeyIn 进行保护,分为两种情况:

(I)如果新父密钥是非对称密钥,如 RSA 或 ECC 密钥,则用新父密钥的公钥加密 encrptionKeyIn,即 CencrptionKeyIn = RSA_OAEP (newParentHandle,encrptionKeyIn)或 ECC_ECDH (newParentHandle ,encrptionKeyIn),并将 sysmetrickey = CencrptionKeyIn;

(II)如果新父密钥是对称密钥,则直接 sysmetrickey = encrptionKeyIn;

(e)执行 outerwrap,由 TPM 产生密钥种子 Seed,用密钥种子 Seed 生成加密密钥和 HMAC 密钥,对 encSensitive 进行加密和 HAMC 运算,得到 dupSensitive 和 outerHMAC;

(f)对 Seed 进行保护,分两种情况:

(Ⅰ)如果新父密钥是非对称密钥,如 RSA 或 ECC 密钥,则用新父密钥的公钥加密 Seed,即 CSeed = RSA_OAEP (newParentHandle, Seed) 或 ECC_ECDH (newParentHandle ,Seed),并将 sysmetricSeed = CSeed;

(Ⅱ)如果新父密钥是对称密钥,则直接 sysmetricSeed = Seed,结束复制过程,转到(4);

(g)对复制密钥既不进行 innerwrap,也不进行 outerwrap,则: dupSensitive = sensitiveArea, outerHMAC = NULL,sysmetrickey = NULL,sysmetricSeed = NULL.

(4) $T_S \rightarrow O_S$: dupSensitive,outerHMAC,sysmetrickey,sysmetricSeed.

(5) $O_S \rightarrow H_S$: dupSensitive,outerHMAC,sysmetrickey,sysmetricSeed .

(6) $H_S \rightarrow H_D$: dupSensitive,outerHMAC,sysmetrickey,sysmetricSeed.

(7) $H_D \rightarrow O_D$: dupSensitive,outerHMAC,sysmetrickey,sysmetricSeed.

(8) $O_D \rightarrow T_D$: dupSensitive,outerHMAC,sysmetrickey,sysmetricSeed .

(9) T_D:执行导入过程.

(a)检查迁移密钥的属性 fixedTPM 和 fixedParent,如果设置不是(0,0)就结束导入过程,转到(10);

(b)检查新父密钥是否为存储密钥,如不是,结束导入过程,转到(10);

(c)根据 sysmetricSeed 参数是否为 NULL 来判断是否进行了 outerwrap,如果为 NULL,转到下一步;否则直接得到 seed 或用新父密钥的私钥解密 sysmetricSeed 得到 seed,按照 symmetricAlg 算法生成 HMAC 密钥 HMACkey 并对 dupSensitive||name 进行 HAMC 运行,得到的值与 outerHMAC 比较,如果不相等,终止导入过程,转到(10);如果相等,则用 seed 生成对称加密密钥 symkey 并解密 dupSensitive 得到 encSensitive ;

(d)根据 sysmetrickey 参数是否为 NULL 来判断是否进行了 innerwrap. 如果为 NULL,则转到(10);否则直接得到 encrptionKeyIn 或用新父密钥的私钥解密 sysmetrickey 得到 encrptionKeyIn,用 symmetricAlg 算法对 encSensitive 进行解密得到 Sensitive 和 name,用 Hash 算法对 Sensitive||name 进行完整性验证,验证通过,则表明迁移成功,转到(10),验证不通过,则表明迁移不成功,转到(10).

(10)结束迁移过程.

3.2 存在的问题

一方面,从 3.1 节可以看出,基于 TPM2.0 密钥复制接口设计的密钥迁移协议存在三个安全问题.

问题 1 该协议缺少源 TPM 和目标 TPM 间的身份认证,导致密钥能够在敌手和 TPM 间迁移:

(1)源 TPM 不能认证新父密钥是否是目标 TPM 的密钥,导致敌手可以用其控制的密钥迁移源 TPM 的密钥,并获得密钥明文;

(2)目标 TPM 不能认证迁移数据是否来自源 TPM,使敌手可以将其控制的密钥迁移到目标 TPM 中.

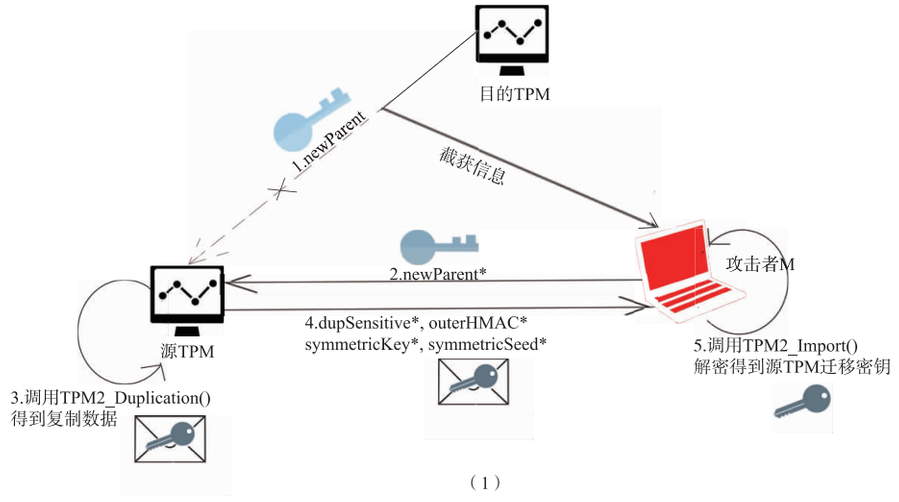
下面我们给出具体的攻击方法及攻击流程图,如图 3 图 4 所示. 其中 M 表示攻击者.

第(1)种情况:

step1. $T_D \rightarrow M$: newParent. publicAera, newparent-Handle, symmetricAlg

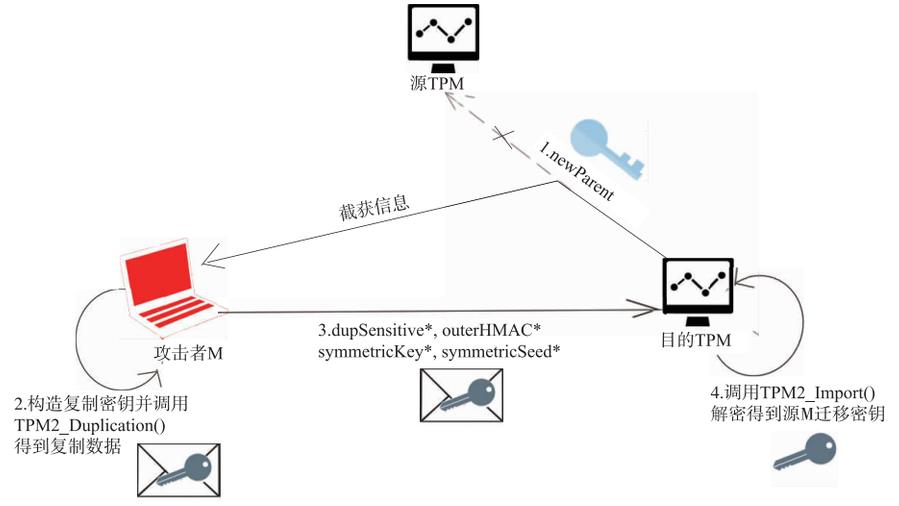
step2. $M \rightarrow T_S$: newParent. publicAera*, newparentHandle*, symmetricAlg*

step3. T_S:调用 TPM2_Duplication (objectHandle,newParentHandle*, encrptionKeyIn, symmetricAlg*)



（1）

图3　第（1）种情况的攻击流程图



（2）

图4　第（2）种情况的攻击流程图

step4. $T_S \rightarrow M$: dupSensitive*, outerHMAC*, symmetricKey*, symmetricSeed*

step5. M: 调用 TPM2_Import（newparentHandle*, symmetricKey*, dupSensitive*, symmetricSeed*）.

step6. End

第(2)种情况:

step1. $T_D \rightarrow M$: newParent. publicAera, newparentHandle, symmetricAlg

step2. M: 调用 TPM2_Duplication（objectHandle*, newParentHandle, encrptionKeyIn*, symmetricAlg）

step3. $M \rightarrow T_D$: dupSensitive*, outerHMAC*, symmetricKey*, symmetricSeed*

step4. T_D: 调用 TPM2_Import（newparentHandle, symmetricKey*, dupSensitive*, symmetricSeed*）

step5. End

问题 2　当复制密钥的属性 encryptedDuplication = 0 且新父密钥的句柄 newParentHandle = TPM_RH_NULL 时,复制接口不能实施 innerwrap 和 outerwrap,迁移密钥将以明文传输而造成泄露,如图 5 所示.

图5　迁移密钥明文信息

问题 3　当新父密钥是对称密钥时,innerwrap 中的对称加密密钥以及 outerwrap 中的密钥种子如何在源 TPM 与目标 TPM 之间安全交换,《TPM-Rev-2.0-Part-1-Architecture-01.38》并没有给出具体的解决办法.

另一方面,3.1 节的密钥迁移协议复制流程比较复杂,其中有几个关键的判断决定复制流程. 首先,密钥能否复制是由复制密钥的属性(fixedTPM,fixedParent)决定;其次,复制过程中是否实施 innerwrap 由复制密钥的 encryptedDuplication 属性决定;第三,复制过程中是否实施 outerwrap 由新父密钥的句柄类型决定;第四,innerwrap 中对称密钥和 outerwrap 中密钥种子的产生和保护交换还依赖新父密钥的密钥类型. 为了保护交换,如果新父密钥是非对称密钥,则 innerwrap 中对称密钥和 outerwrap 中密钥种子在源 TPM 方产生;如果新父密钥是对称密钥,而复制密钥是非对称密钥,则 innerwrap 中对称密钥和 outerwrap 中密钥种子在目的方产生;如果新父密钥和复制密钥均是对称密钥,则 innerwrap 中对称密钥和 outerwrap 中密钥种子既可以在源 TPM 方产生,也可以在目的 TPM 方产生,但如何保护交换还没有比较好的方法. 由此可见,不同的属性值决定了不同的复制流程,在所有的流程中,部分流程的输出结果是存在安全隐患的. 特别地,为了保护交换 innerwrap 中对称密钥和 outerwrap 中密钥种子,此对称密钥和密钥种子的产生方究竟在源 TPM 方还是目的 TPM 方具有不确定性,因此这些复制流程并不统一.

4 认证密钥和证书

4.1 迁移认证密钥—MAK

迁移认证密钥的主要作用是和复制接口一起配合进行密钥迁移,是一个非对称密钥,基本属性如表 3.

表 3 迁移认证密钥的基本属性

Sign	Decrypt	Restricted	功能
1	1	1	仅用密钥迁移, 能够实施 Sign 和 Decrypt 操作

其密钥结构如下:

```
Public Area = {
    type = asymmetric
        nameAlg = RSA
        objectAttributes = {sign = 1, decrypt = 1, restricted = 1, fixTPM = 1, fixParent = 1}
        authPolicy = Default
    parameters = TPMS_RSA_PARMS
  unique = Publickey_MAK
}
Sensitive Area = {
        SensitiveType = TPMI_ALG_PUBLIC
        AuthValue = Default
        SeedValue = NULL
        Sensitive = Privatekey_MAK
}
```

MAK 由 SRK 直接保护,是一个不可复制密钥,也是存储密钥. MAK 在层次保护体系中的位置如图 6 所示.

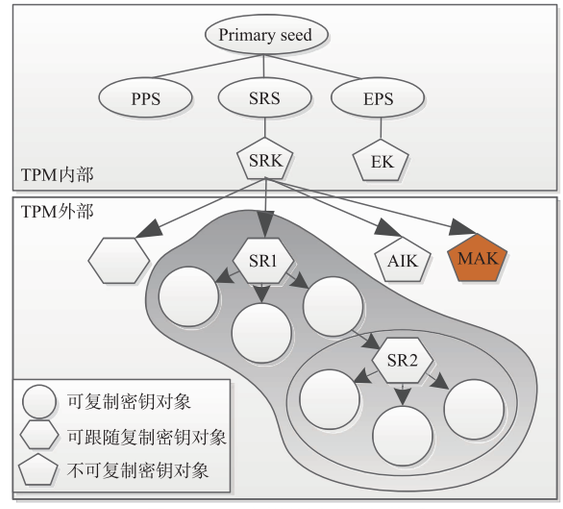


图6 MAK密钥的层次保护体系

4.2 迁移认证证书—$Cert_{MAK}$

在 TPM2.0 中,如果需要为某一密钥对象颁发证书,则要求在创建该密钥对象时为该密钥对象时生成 TPMS_CREATION_DATA 类型的 creationData 数据,TPMS_CREATION_DATA 的定义如下:

```
Typedef Structure{
        TPML_PCR_SELECTION pcrSelect
        TPM2B_DIGEST pcrDigest
        TPMA_LOCALITY locality
        TPM_ALG_ID parentNameAlg
        TPM2B_NAME parentName
        TPM2B_NAME   parentQualifiedName
        TPM2B_DATA outsideInfo
} TPMS_CREATION_DATA
```

从 TPMS_CREATION_DATA 可以看出,creationData 中包含了平台度量值、PCR 值及其签名、密钥对象在层次结构模型的位置、父密钥支持的密钥算法、层次名字和一些外部信息.

将 MAK 密钥对象的 creationData(可以加上 EK 证书)发送到某一证书颁发机关(PCA)申请相应的证书. PCA 对 creationData(包括 EK)进行验证,验证通过后颁发一个 X.509v3 的证书,如下.

```
Cert_MAK : : = SEQUENCE {
  tbsCertificate TBSCertificate,
  signatureAlgorithm AlgorithmIdentifier,
  signatureValue BITSTRING
}
```

```
TBSCertificate::= SEQUENCE {
    version v3,
    serialNumberCertificateSerialNumber default,
    signatureAlgorithmIdentifier default,
    issuerName default,
    validity default,
    subjectName MAKCertification,
    subjectPublicKeyInfo,SubjectPublicKeyInfo,
    issuerUniqueID default,
    subjectUniqueID default,
    extensions Extension,
}
AlgorithmIdentifier::= SEQUENCE {
    algorithm OBJECT IDENTIFIER,
    parameters ANY DEFINED BY algorithm OPTIONAL
    }
Extension::= SEQUENCE {
    extnID OBJECT IDENTIFIER,
    critical BOOLEAN DEFAULT FALSE,
    extnValue OCTET STRING
    nameAlg ECC|RSA
    }
```

值得注意的是,该证书虽是一个普通的 X.509 证书,但该证书不仅可以用来进行身份认证,还可以用来对会话密钥加密,因此,在扩展项中增加了源 TPM 和目标 TPM 里都支持的非对称加密算法.

5　MAK 和复制接口的新密钥迁移协议—MDKMP

针对上述问题,本节将基于 MAK 和密钥复制接口设计新的密钥迁移协议.新的密钥迁移协议首先需要对 TPM 初始化,目的是在 TPM 中生成存储于 SRK 之下的非对称密钥 MAK 并向可信第三方申请证书,然后再对源 TPM 中的复制密钥进行迁移.值得注意的是,此迁移不是直接将复制密钥迁移到目标 TPM 的实际父密钥下,而是分两个阶段:第一阶段是将复制密钥迁移到目标 TPM 的 MAK 下,这个阶段是在源 TPM 和目的 TPM 之间进行,我们称为外部迁移;第二阶段是将已迁移到 MAK 下的复制密钥迁移到实际的父密钥下,这个阶段是在目标 TPM 内部进行,我们称为内部迁移.具体的迁移流程如图 7 所示,其中 Mig 表示待迁移密钥,NP 表示真正的新父密钥.

5.1　初始化

初始化的主要作用是 TPM 平台产生 MAK 密钥对,并向可信第三方申请证书.经过初始化后 TPM 不仅得到了 MAK 密钥和 MAK 证书,同时还进行了自身合法性的认证.无论是源 TPM 还是目标 TPM 均需要进行初始化.我们用 $Cert_{CA}$ 表示可信第三方 CA 的证书,(PubMAK,PriMAK)表示 MAK 的公钥对,EK 表示 TPM 的背

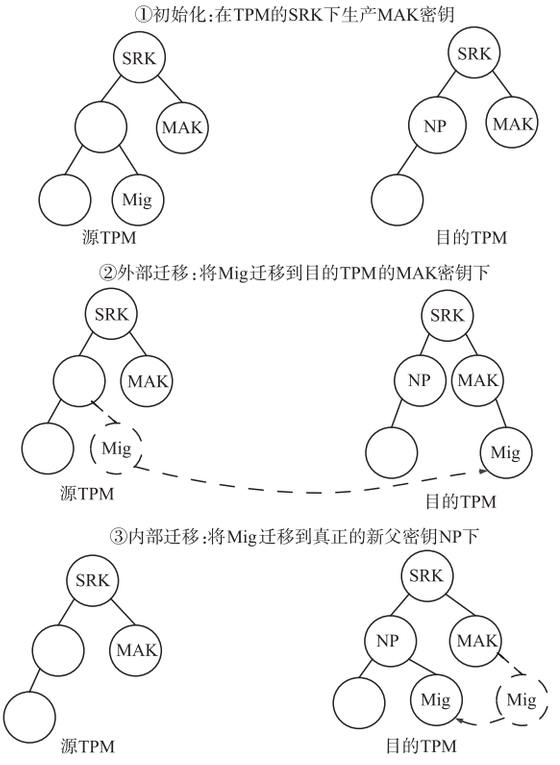


图7　整体迁移流程图

书证书,具体流程如下,其流程图如图 8 所示.

(1) TPM→CA:Enc(k,[PubMAK ‖ creationData ‖ N_0 ‖ Sign(PubMAK ‖ creationData ‖ N_0, PriMAK) ‖ EK]),Enc($Cert_{CA}$,k)

(2) CA→TPM:Enc(k ,CertMAK ‖ N_0)

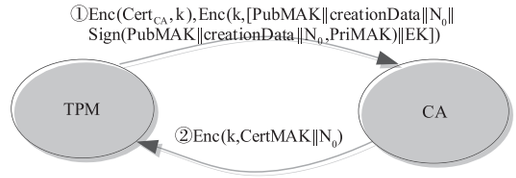


图8　初始化阶段流程图

下面我们对上述每一步进行详细解释,具体如下.

步骤一　TPM 首先调用 TPM2_Create()创建 MAK 公私钥对 MAK=(PubMAK,PriMAK),并将 PriMAK 由 SRK 保存;然后产生一个 TPMS_CREATION_DATA 类型的数据 creationData 和随机数 N_0,并签名 Sign(creationData ‖ N_0,PriMAK);第三,产生会话密钥 k,并计算 Enc(k,[PubMAK ‖ creationData ‖ N_0 ‖ Sign(PubMAK ‖ creationData ‖ N_0,PriMAK) ‖ EK]),第四,用 CA 的证书对会话密钥加密;最后,将 Enc($Cert_{CA}$,k) 和 Enc(k,[PubMAK ‖ creationData ‖ N_0 ‖ Sign(PubMAK ‖ creationData ‖ N_0, PriMAK) ‖ EK])一起传给可信第三方 CA;

步骤二 CA 首先用 $Cert_{CA}$ 证书对应的私钥解密 $Enc(Cert_{CA}, k)$，得到会话密钥 k；然后解密 $Enc(k, [PubMAK \| creationData \| N_0 \| Sign(PubMAK \| creationData \| N_0, PriMAK) \| EK])$，得到 PubMAK、creationData、$N_0$、$Sign(PubMAK \| creationData \| N_0, PriMAK)$ 和 EK. 然后 CA 验证 EK 和 $Sign(PubMAK \| creationData \| N_0, PriMAK)$，如果验证都通过，表明这些数据来自于一个合法的 TPM；第三，CA 产生 MAK 证书 CertMAK；最后将 $Enc(k, [CertMAK \| N_0])$ 传递给 TPM，TPM 解密获得证书 CertMAK，然后验证随机数 N_0，如果验证通过表明申请证书已成功.

5.2 第一阶段：外部迁移

第一阶段的主要功能是将源 TPM 中的复制密钥迁移到目标 TPM 的 MAK 密钥下. 我们用标识符 outer_Migrate_Request 表示外部迁移请求，$CertMAK_S$ 表示源 TPM 的 MAK 证书，newParent_publicArea 表示实际父密钥的公钥，MAK_D_publicArea 表示目标 TPM 的 MAK 公钥，dupSensitive 表示复制密钥的 Blob 密文，outerHMAC 表示 outerwrap 的消息码，CencryptionKeyout 表示 innerwrap 的对称密钥密文，CSeed 表示 outerwrap 的密钥种子密文，encRng 表示随机数密文，具体的协议流程如下，其流程图如图 9 所示.

(1) $T_D \rightarrow T_S$: outer_Migrate_Request, $Enc(k, [newParent\_publicArea \| MAK_D\_publicArea \| N_{T_D} \| Sign(newParent\_publicArea \| MAK_D\_publicArea \| N_{T_D}, PriMAK_D)])$, $Enc(CertMAK_S, k)$

(2) $T_S \rightarrow T_D$: dupSensitive, outerHMAC, CencryptionKeyout, CSeed, encRng

(3) $T_D \rightarrow T_S$: encRng

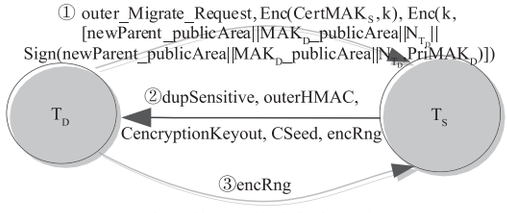


图9 外部迁移阶段流程图

下面，我们对上述每一步进行详细解释，具体如下.

步骤一 T_D 首先产生会话密钥 k，并计算 $Enc(k, [newParent\_publicArea \| MAK_D\_publicArea \| N_{T_D} \| Sign(newParent\_publicArea \| MAK_D\_publicArea \| N_{T_D}, PriMAK_D)])$，然后用源 TPM 的 $CertMAK_S$ 加密会话密钥 k，并将 outer_Migrate_Request, $Enc(k, CertMAK_S)$, $Enc(k, [newParent\_publicArea \| MAK_D\_publicArea \| N_{T_D} \| Sign(newParent\_publicArea \| MAK_D\_publicArea \| N_{T_D}, PriMAK_D)])$ 一起传递给 T_S.

步骤二 在本步骤中，T_S 将进行复杂判断和运算，具体如下.

(a) 首先 T_S 用 MAK 私钥解密 $Enc(k, CertMAK_S)$ 得到会话密钥 k，用 k 解密 $Enc(k, newParent\_publicArea \| MAK_D\_publicArea \| N_{T_D} \| Sign(newParent\_publicArea \| MAK_D\_publicArea \| N_{T_D}, PriMAK_D))$ 得到 newParent_publicArea、MAK_D_public Area 和随机数 N_{T_D} 以及签名值. 然后利用 $CertMAK_D$ 对签名值进行验证，验证通过则表明消息来自于 T_D.

(b) 然后 T_S 将根据复制密钥的(fixedTPM, fixedParent, encryptedDuplication) 和新父密钥的句柄 newParentHandle 类型，判断该复制密钥能否迁移. 有如下情况 T_S 会直接终止迁移过程，所有的返回值都置 NULL.

当 fixedTPM! =0 或 fixedParent! =0 时，T_S 不进行任何运算，直接 dupSensitive = NULL, outerHMAC = NULL, CencryptionKeyout = NULL, CSeed = NULL, encRng.

(c) 如果不是步骤(b)中的情况，则 T_S 将目的 TPM 的 MAK 作为新父密钥，进行密钥复制并进行迁移. 由于 MAK 密钥是非对称密钥，其句柄类型不是 TPM_RH_NULL. 因此，T_S 将仅根据复制密钥的 encryptedDuplication 决定计算流程，一共有两种情况：

当 encryptedDuplication = 1 时，T_S 进行的具体运算如下：

(I) 产生 innerwrap 需要的对称加密密钥 encryptionKeyIn(可 T_S 的 O_S 输入)；

(II) 计算 innerIntegrity = $\mathbf{H}_{object.\ nameAlg}$(object. sensitive ‖ object. name)；

(III) 计算 encSensitive = $\mathbf{CFB}_{object.\ symAlg}$(encryptionKeyIn, 0, innerIntegrity ‖ object. sensitive)；

(IV) 根据 AlgParameter，计算 CencryptionKeyIn = RSA_OAEP(CertMAK$_D$, encryptionKeyIn) 或 CencryptionKeyIn = ECC_ECDH (CertMAK$_D$, encryptionKeyIn)；

(V) 产生 outerwrap 需要的密钥种子 seed；

(VI) 计算 symKey = **KDFa**(AlgParameter, seed, "STORAGE", Name, NULL, bits)；

(VII) 计算 dupSensitive = $\mathbf{CFB}_{AlgParameter}$(symKey, 0, encSensitive)；

(VIII) 计算 encRng = $\mathbf{CFB}_{AlgParameter}$(symKey, 0, [$N_{T_D} \| N_{T_S}$])；

(IX) 计算 HMACkey = **KDFa**(AlgParameter, seed, "INTEGRITY", NULL, NULL, bits)；

(X) 计算 outerHMAC = $\mathbf{HMAC}_{npNameAlg}$(HMACkey, dupSensitive ‖ objecthandle -> Name)；

(XI) 根据 seed，计算 Cseed = RSA_OAEP (Cert-

MAK_D,seed)或 CencryptionKeyIn = ECC_ECDH (Cert-MAK_D,seed).

其中,RSA_OAEP()和 ECC_ECDH 是《TPM-Rev-2.0-Part-1-Architecture-01.38》推荐的两种非对称保护算法.以上过程实际上是调用 TPM2_Duplicate (objectHandle, newParentHandle, encryptionKeyIn, symmetricAlg)进行密钥复制,对复制密钥实施 innerwrap 和 outerwrap.

当 encryptedDuplication = 0 时,Ts 进行的具体运算如下:

(I)产生 outerwrap 需要的密钥种子 seed;

(II)计算 symKey = **KDFa** (AlgParameter, seed, "STORAGE",Name,NULL ,bits);

(III)计算 dupSensitive = $\mathbf{CFB}_{\text{AlgParameter}}$(symKey,0, encSensitive);

(IV)计算 encRng = $\mathbf{CFB}_{\text{AlgParameter}}$(symKey,0,[N_{T_D} ‖ N_{T_S}]);

(V)计算 HMACkey = **KDFa** (AlgParameter, seed, "INTEGRITY",NULL,NULL,bits);

(VI)计算 outerHMAC = **HMAC**npNameAlg (HMACkey,dupSensitive ‖ objecthandl e - > Name);

(VII)根据 seed,计算 Cseed = RSA_OAEP (Cert-MAK_d,seed)或 CencryptionKeyIn = ECC_ECDH (Cert-MAK_d,seed).

以上过程实际上是调用 TPM2_Duplicate (objectHandle, newParentHandle, encryptionKeyIn, symmetricAlg)进行密钥复制,对复制密钥实施 outerwrap.

步骤三　T_D接收到 T_S传来的 dupSensitive、outerHMAC、CencryptionKeyout、CSeed 和 encRng,进行如下计算.

(a)首先检查 CSeed 是否为 NULL.如果为 NULL,就直接终止导入过程.

(b)用新父密钥的私钥解密 CSeed,得到 Seed.

(c)用同样的算法生成 HMACkey 和 symKey.

(d)用 symKey 解密 encRng,验证随机数 N_{T_D}和 N_{T_S},再用 HMACkey 验证 outerHMAC.如果验证都通过,用 symKey 对 dupSensitive 进行解密得到 encSensitive.如果验证不通过,就直接终止导入过程.

(e)用新父密钥的私钥解密 CencryptionKeyout 得到 encryptionKeyout,用 encryptionKeyout 解密 encSensitive 得到 innerIntegrity 和 Sensitive,用同样的 **H** 算法验证 object.sensitive ‖ object.name 的完整性.如果验证通过,则外部迁移成功.

5.3　第二阶段:内部迁移

第二阶段的主要功能是将第一阶段迁移到目标 TPM 的 MAK 密钥下的复制密钥迁移到实际的新父密钥下.由于该阶段迁移均在目标 TPM 内部进行,不需要在外部交换复制密钥、inerwrap 对称加密密钥和 outerwrap 密钥种子,因此,可以不考虑复制密钥、inerwrap 对称加密密钥和 outerwrap 密钥种子的安全保护.我们用标识符 inner_Migrate_Request 表示内部迁移的请求,具体协议流程如下,其流程图如图 10 所示.

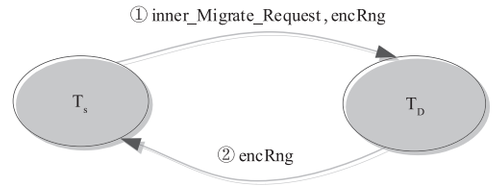


图10　内部迁移阶段流程图

(1)$T_S \rightarrow T_D$:inner_Migrate_Request,encRng

(2)$T_D \rightarrow T_S$:encRng

下面,我们对上述每一步进行详细解释,具体如下.

步骤一　T_S向 T_D发起内部迁移请求 inner_Migrate_Request,并沿用第一阶段的随机数 encRng.

步骤二　T_D首先调用 TPM2_Duplication (objectHandle, newParentHandle, encrptionKeyIn, symmetricAlg)进行密钥复制,输出 dupSensitive,outerHMAC,encryptionKeyout,Seed,Rng.其间 inerwrap 对称加密密钥、outerwrap 密钥种子和随机数均不需要保护.然后调用 TPM2_Import (newparentHandle, encryptionKeyout, dupSensitive,Seed)导入复制密钥.如果导入成功,则 T_D向 T_S返回 encRng,T_S解密 encRng,验证随机数,如果验证通过,表明内部迁移成功.

6　密钥迁移协议的特点及安全性分析

本节对该密钥迁移协议进行特点分析和安全分析.

6.1　特点分析

(1)一致性.对于 TPM2.0 的密钥迁移,TCG 定义了两个接口详情见本文第 2 节内容,本文提出的新协议的整个迁移流程符合 TCG 定义的规范和标准.

(2)统一性.新父密钥可以是对称密钥也可是非对称密钥,由此解除了新父密钥只能是非对称密钥的限制,增加了协议的灵活性.

(3)创新性.本方案最大的特点之一,就是在 TPM 新增了一类可选证书—CertMAK,只用于 TPM 迁移和密钥迁移,扩展了 TPM 的证书体系.此外,利用 MAK 证书对消息进行签名,使双方间的身份验证变得简单高效.

(4)认证性.本文使用 CertMAK 对数据,消息进行签名以保证消息来源的可靠性,从而实现源和目的

TPM 之间的相互认证.

(5)简单性. 在第 2 节中不管是 innerwarp 还是 outerwarp 都需要考虑迁移密钥和新父密钥是对称还是非对称密钥,由此衍生出 4 种不同的组合也就需要 4 种方案来保护加密密钥传输. 而加入 MAK 密钥后,协议变得十分简单,因为 MAK 是非对称的,因此不管是 innerwarp 还是 outerwarp 都无需考虑迁移密钥和新父密钥的种类.

6.2 安全性分析

我们提出的密钥迁移协议具有机密性、完整性、抗中间人攻击和抗重放攻击,因而具有较高的安全性. 对本文提出的方法进行安全性分析,具体有以下四点.

(1)具有机密性

在协议的外部迁移阶段中,步骤 1 中使用会话密钥 k 加密交互信息(newParent_publicArea ‖ MAK_D_publicArea ‖ N_{T_D} ‖ Sign(newParent_publicArea ‖ MAK_D_publicArea ‖ N_{T_D}, $PriMAK_D$)),然后利用 $CertMAK_S$ 加密 k. 这样即使信息被截取,攻击者会由于没有 k 而无法获得信息的具体内容,由此确保 T_D 和 T_S 间的通信安全. 步骤 2 执行 innerwrap 和 outerwrap 时用对称加密密钥 encryptionKeyIn 对 sensitive ‖ name 的哈希值 innerIntegrity 以及复制密钥的 sensitive 进行加密,用密钥种子 seed 作为 KDFa( )的参数产生对称加密密钥对 innerwrap 阶段得到的 encSensitive 进行加密,并用 $PubMAK_D$ 加密 encryptionKeyIn 和 seed. 如果信息被截取,一方面,攻击者会由于没有 $PriMAK_D$,从而无法获得 seed ,因而也无法获得 encSensitive. 另一方面,攻击者也无法获得 encryptionKeyIn,因为 encryptionKeyIn 由 TPM 内部产生或由 TPM 的合法用户输入,因而攻击者无法解密获得迁移密钥的 sensitive,由此确保复制密钥的机密性. 在协议的内部迁移阶段,由于该阶段密钥的复制迁移主要是在 TPM 内部进行,不存在安全威胁,也就保证了机密性.

(2)具有完整性

在协议的外部迁移阶段,步骤 1 中使用 $PriMAK_D$ 对消息 newParent_publicArea ‖ MAK_D_publicArea ‖ N_{T_D} 进行签名. 这样如果消息被截获,攻击者也无法篡改或伪造数据,由此保证了 T_D 和 T_S 间通信的完整性. 步骤 2 中执行 innerwarp 时利用 Hash 算法对 object. sensitive ‖ object. name 进行摘要计算,执行 outerwarp 时利用 HMAC 算法对 dupSensitive ‖ objecthandle→Name 进行计算得到消息码 outerHMAC. 即使攻击者窃听到消息摘要,仍然无法篡改或伪造迁移密钥的各种属性数据,因而确保迁移消息的完整性.

(3)具有抗中间人攻击

中间人攻击是一种通过修改或者伪装发送消息而达到攻击目的手段. 在协议的外部迁移阶段,步骤 1 中 T_D 向 T_S 发送的信息使用了 T_S 的 MAK 证书 $CertMAK_S$ 对会话密钥 k 进行加密,只能用 T_S 的 $PriMAK_S$ 解密 Enc($CertMAK_S$, k)得到 k,由此保证消息正确传输到 T_S 且只能被 T_S 解密. 同时用 $PriMAK_D$ 对 newParent_publicArea ‖ MAK_D_publicArea ‖ N_{T_D} 签名,只有用 T_D 的证书 $CertMAK_D$ 才能验证此签名,以此表明此消息来自于 T_D. 通过这种方式实现了 T_D 与 T_S 间的相互认证,因此攻击者无法对此消息进行篡改和伪造,防止了中间人攻击. 步骤 2 中 T_S 返回的数据是经过 innerwrap 及 outerwrap 运算得到的,其中在 innerwrap 中使用 $CertMAK_D$ 对 encryptionKeyIn 加密,在 outerwrap 中使用 $CertMAK_D$ 对 seed 加密. 当 T_D 接收到消息后只有使用 $PriMAK_D$ 才能解密出 encryptionKeyIn 和 seed,以此保证此消息只有 T_D 才能进行解密,因此攻击者无法对此消息进行篡改和伪造,防止了中间人攻击. 而在协议的内部迁移阶段,由于该阶段密钥的复制迁移主要是在 TPM 内部进行,不存在安全威胁,也就能抗中间人攻击.

(4)具有抗重放攻击

在协议的外部迁移阶段,步骤 1 中 T_D 向 T_S 发送的信息中包含现时 N_{T_D}. 步骤 2 中 T_S 返回 encRng,而 encRng 包含 N_{T_D} 和 N_{T_S}. 步骤 3 中 T_D 返回的消息中也包含 N_{T_S}. 因此 T_D 可以通过对比 N_{T_D} 来确保消息不是重放消息,同样 Ts 也可以通过对比 N_{T_S} 来确保消息不是重放消息. 在协议的内部迁移阶段,内部迁移阶段沿用了外部迁移阶段 encRng,因此 T_D 和 Ts 都可以通过对比 N_{T_S} 或 N_{T_D} 来确保消息不是重放消息.

综上,整个协议具有完整性,机密性并且是可以抗中间人攻击和重放攻击的.

7 密钥迁移协议实验原型分析

要将复制密钥迁移到新父密钥下,首先需要进行初始化:TPM 平台产生 MAK 密钥对,并向可信第三方申请证书. 然后,目的 TPM 将自己的 MAK_D 等信息发送给源 TPM. 紧接着源 TPM 根据复制密钥的属性(fixedTPM,fixedParent)判断是否能够迁移,然后再根据 encryptedDuplication 的取值判断是否需要进行 innerwarp,并将 innerwarp,outerwrap 后的数据发送给目的 TPM,随后目的 TPM 解密数据并加载复制密钥到自己的 MAK 下,完成外部迁移. 最后,目的 TPM 再进行一次内部迁移,将复制密钥迁移到实际的新父密钥下,即可完成迁移.

7.1 实验环境

目前市面上已出现 TPM2.0 芯片,经过深入的咨询我们发现在国内使用 TPM2.0 芯片受到一定的限制. 因

此这里我们使用微软的 TPM2.0 模拟器以及 TSS.net 进行实验. 本实验共使用 2 台计算机,其中一台既做 target TPM 又充当 CA,剩下一台充当 source TPM. 这里我们用 OpenSSL 的 CA 机关,作为可信第三方 CA. 具体环境配置如表 4 所示,其中 target TPM 与 CA 所在主机的配置完全一致.

表 4　实验环境相关配置表

	source TPM	target TPM	CA
CPU	Intel(R) Core (TM) i5-3210M CPU @2.50GHZ	Intel(R) Core (TM) i7-7500U CPU @2.70GHZ	Intel(R) Core (TM) i7-7500U CPU @2.70GHZ
内存	4GB	8GB	8GB
OS	Windows 7 32bit	Windows 10 64bit	Windows 10 64bit
IP 地址	192.168.0.109/ 169.254.8.81	192.168.0.101/ 169.254.12.45	192.168.0.101/ 169.254.12.45
子网掩码	255.255.255.0/ 255.255.0.0	255.255.255.0/ 255.255.0.0	255.255.255.0/ 255.255.0.0
Openssl 版本	Openssl_1.0.0	Openssl_1.0.0	Openssl_1.0.0
TPM 模拟器	TSS.MSR v2.0 TPM2 Simulator	TSS.MSR v2.0 TPM2 Simulator	TSS.MSR v2.0 TPM2 Simulator
TSS 版本	TSS.MSR-master	TSS.MSR-master	TSS.MSR-master

7.2　实验结果

(1)初始化阶段的执行过程:初始化过程主要完成 TPM 产生自己的 MAK,并向可信第三方申请证书. 由于目的 TPM 的初始化与源 TPM 初始化过程完全相同,这里以目的 TPM 的初始化为例. 目的 TPM 与 CA 间的初始化阶段流程如图 11 ~ 12 所示.

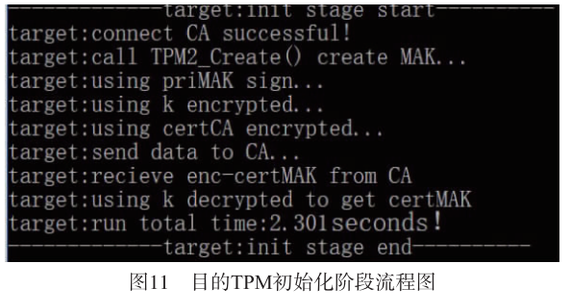


图11　目的TPM初始化阶段流程图

(2)外部迁移阶段:外部迁移阶段的主要功能是将源 TPM 中的复制密钥迁移到目标 TPM 的 MAK 密钥下. 根据 4.2 节,分以下三种情况.

情况 1　当 fixedTPM! =0 或 fixedParent! =0 时,不能进行迁移. 图 13 ~ 14 为目的 TPM 和源 TPM 间的执行流程图.

情况 2　当 fixedTPM = 0, fixedParent = 0, encrypted-Duplication = 1, newParentHandle! = TPM_RH_NULL 时,既进行 innerwrap 又进行 outerwrap. 目的 TPM 和源 TPM 间具体的流程如图 15 ~ 16 所示.

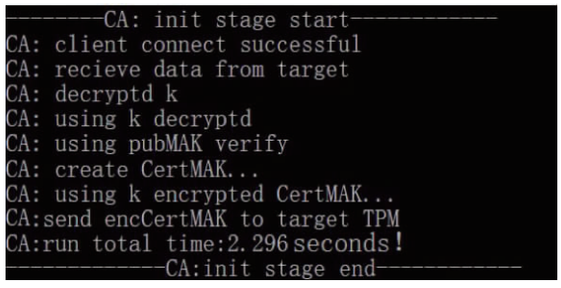


图12　CA初始化阶段流程图

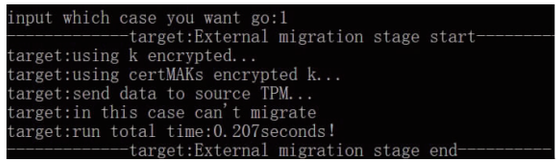


图13　目的TPM外部迁移流程图

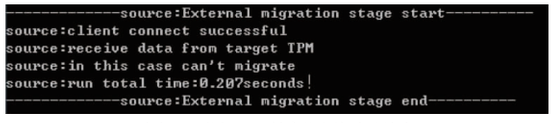


图14　源TPM外部迁移流程图

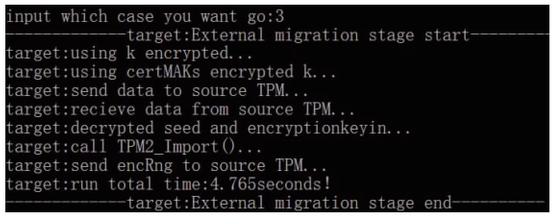


图15　目的TPM外部迁移流程图

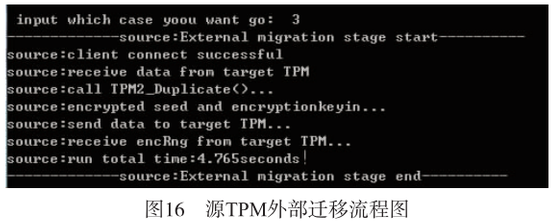


图16　源TPM外部迁移流程图

情况 3　当 fixedTPM = 0, fixedParent = 0, encrypted-Duplication = 0, newParentHandle! = TPM_RH_NULL 时,只进行 outerwrap. 目的 TPM 和源 TPM 间具体的流程如图 17 ~ 18 所示.

(3)内部迁移阶段:内部迁移主要任务是将复制密钥迁移到实际的新父密钥下. 目的 TPM 和源 TPM 间具体的流程如图 19 ~ 20 所示.

7.3　性能分析

(1)初始化阶段的执行时间如表 5 所示.

```
input which case you want go:4
------------target:External migration stage start----------
target:using k encrypted...
target:using certMAKs encrypted k...
target:send data to source TPM...
target:recieve data from source TPM...
target:decrypted seed...
target:call TPM2_Import()...
target:send encRng to source TPM...
target:run total time:4.119seconds!
------------target:External migration stage end----------
```

图17 目的TPM外部迁移流程图

```
 input which case yoou want go:  4
------------source:External migration stage start----------
source:client connect successful
source:receive data from target TPM
source:call TPM2_Duplicate()...
source:encrypted seed ...
source:send data to target TPM...
source:receive encRng from target TPM...
source:run total time:4.114seconds!
------------source:External migration stage end----------
```

图18 源TPM外部迁移流程图

```
------------target:Inner migration stage start----------
target: receive data from source TPM
target:call  TPM2_Duplicate and TPM2_Import()...
target:send encRng to source TPM...
target:run total time:2.166seconds!
------------target:Inner migration stage end----------
```

图19 目的TPM内部迁移流程图

```
------------source:Inner migration stage start----------
source:client connect successful
source:send inner_Migrate_Request to target TPM...
source:receive encRng from target TPM...
source:run total time:2.165seconds!
------------source:Inner migration stage end----------
```

图20 源TPM内部迁移流程图

表 5 初始化阶段执行时间表

	源 TPM	目的 TPM	CA
执行时间	2.301s	2.301s	2.296s

(2)外部迁移阶段执行时间如表 6 所示.

表 6 外部迁移阶段执行时间表

	源 TPM	目的 TPM
情况 1	0.207s	0.207s
情况 2	4.765s	4.765s
情况 3	4.114s	4.119s

(3)内部迁移执行时间如表 7 所示.

表 7 内部迁移阶段执行时间表

	源 TPM	目的 TPM
执行时间	2.165s	2.166s

具体的时间耗时性能图如图 21 所示.

从图 21 可以看出,外部迁移情况 1 耗时最短,这是因为在此情况下不满足可迁移条件,无法进行迁移,所以耗时最短. 在可正常迁移情况下,外部迁移情况 3 比情况 2 少进行了 innerwarp 所以情况 2 的耗时长于情况 3. 而内部迁移过程不用进行加密密钥和密钥种子的传输,源 TPM 与目的 TPM 间也不需要交互,所以内部迁移耗时比外部迁移短.

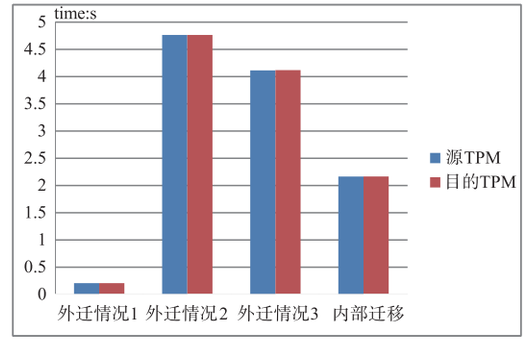


图21 时间耗时性能图

8 相关工作

在 TPM 密钥迁移方面,到现在为止取得了较多研究成果. TCG 组织一直是这一工作积极的主导者和推动者,TCG 最初在 TPM1.1 规范的密钥迁移模块中提供了

TPM_AuthorizeMigratinKey( )

TPM_CreateMigrationBlob( )

TPM_ConvertMigrationBlob( )

等三个 API 接口用于迁移密钥. TPM1.1 的密钥迁移模块优点是简单,但弱点是迁移源和目标之间缺少认证,敌手很容易冒充目标 TPM 而获得迁移密钥的敏感信息. TPM1.2 对迁移模块做了重大改进,引进了认证迁移概念. 在认证迁移下,用户可以指定可信第三方迁移权威(Migration Authorites,MA)并保证如果没有 MA 的允许,不可以把密钥迁移到特定的平台. TPM1.2 关于认证迁移密钥(CMK,Certifiable-migration key)的 API 包括:

TPM_AuthorizeMigratinKey( )

TPM_CMK_ApproveMA( )

TPM_CMK_CreateKey( )

TPM_CMK_CreateTicket( )

TPM_CMK_Create-Blob( )

TPM_CMK_ConvertMigration( )

等共六个接口. TPM1.2 的密钥迁移模块优点是安全,但缺点是复杂,每次迁移均需要可信第三方 MA 参与,效率低. TPM2.0 取消了 TPM1.2 的密钥迁移接口,并使用新的迁移接口,即:TPM2_Duplicate( )和 TPM2_Import( ). 由于 TPM2.0 改变了密钥属性的表示方式,一个密钥对象是否可以被复制由 fixedTPM 和 fixedParent 两个属性组合来决定,当(fixedTPM = 0,fixedParent = 0)时候,该密钥对象可以被复制实现迁移,当(fixedTPM = 0,fixedParent = 1)该密钥对象可以跟随父密钥一起被复制实现迁移. TPM2.0 使用 TPM2_Duplicate( )进行密钥复制时,需要对被复制密钥进行 innerwrap 或 outerwrap 或既进行 innerwrap 又进行 outerwrap 来保证被复制密钥的机密性、完整性以及认证性. TPM2.0 的密钥迁移模块优点是简单、高

效、灵活而且基本安全,可不需要可信第三方参与. 但缺点是容易出现配置错误,导致不安全.

在 TPM1.1 和 TPM 1.2 密钥迁移机制的分析方面,文献[13]用一阶逻辑语言建立 TPM API 的形式化模型,并对 TPMAPI 进行了全面的逻辑推理分析,其中对密钥迁移 API 的分析指出 TPM1.1 的弱点在于迁移目标由源 TPM 的 owner 指定,目标 TPM 并不参与迁移,目标 TPM 在接收可迁移密钥时,可迁移密钥有可能已经泄漏. 因此具有较大的安全隐患. 文献[14]应用 π 演算对 TPM 进行形式化建模,并使用自动定理证明工具 ProVerif 验证其安全属性. 作者分析了 TPM CMK(Certifiable Migratable Key)的 RESTRICT_ MIGRATE 迁移模式,分析结果表明:若作为第三方的迁移权威(Migration Authority,MA)用软件处理迁移数据,则敌手能获得被迁移密钥的私钥. 作者建议 TPM 规范强制要求 MA 使用 TPM 代替软件处理迁移数据. 文献[15]对 TPM 可迁移密钥的安全性进行了分析,指出 TPM 提供密钥迁移机制的同时,降低了可迁移密钥的安全保护强度,敌手能够利用 TPM 的密钥迁移类接口和密钥加载接口破坏 TPM 可迁移密钥的安全性.

在 TPM2.0 密钥迁移机制的分析方面,文献[16]建立了 TPM2.0 保护存储 API 的抽象模型,并利用类型系统证明了 TPM2.0 保护存储的安全性. 证明结果表明,TPM2.0 保护存储中的密钥复制类接口是安全的. 文献[17]对 TPM2.0 密钥管理 API 的安全性进行了形式化分析,证明了密钥存储和使用类接口能够保证 TPM 不可迁移密钥的安全性,并发现了针对密钥复制类接口的两种攻击. 作者提出了该类接口的改进方案,并证明了利用改进的接口实施密钥复制能够保证被复制密钥的安全性. 文献[18]分析了 TPM2.0 密钥复制相关流程,对于其中存在的密钥隐私泄露问题进行了改进. 在用户不安全复制传输情形下,从 TPM 管理者的角度出发提出了一套基于 TPM 自身的加密传输协议. 通过利用 TPM 自身产生安全密钥,对未受保护的用户敏感数据进行加密,并通过签名的方式保障传输的可靠性.

另外,对于我国 TCM 芯片,文献[19]指出由该芯片的密钥迁移模块实现的密钥迁移协议存在两个问题:对称密钥不能作为被迁移密钥的新父密钥,违背了 TCM 的初始设计思想;缺少交互双方 TCM 的相互认证,导致源 TCM 的被迁移密钥可以被外部敌手获得,并且敌手可以将其控制的密钥迁移到目标 TCM 中. 针对上述问题,作者提出两个新的密钥迁移协议:协议 1 遵循 TCM 目前的接口规范,以目标 TCM 的 PEK(Platform Encryption Key)作为迁移保护密钥,能够认证目标 TCM,并允许对称密钥作为新父密钥;协议 2 简单改动了 TCM 接口,源 TCM 和目标 TCM 进行 SM2 密钥协商,得到的会话密钥作为迁移保护密钥,解决了上述两个问题,并且获得了前向安全属性. 最后,使用形式化分析方法对上述协议进行安全性分析,分析结果显示,协议满足正确性和预期的安全属性.

9　总结

本文首先对 TPM2.0 密钥迁移协议进行了研究和分析,通过进一步研究发现,基于密钥复制接口的密钥迁移协议至少存在三个问题,并针对三个问题,首先在 TPM 中增加了 CertMAK 证书,然后基于 MAK 证书和复制接口设计新的 TPM 密钥迁移协议 MDMKP. 安全分析表明,该协议不仅能很好地解决前面那三个问题,而且能够统一协议流程,降低复杂度,保证复制密钥不会泄露. 最后,通过模拟实验验证显示,该协议满足正确性和预期的安全属性.

本文为 TPM2.0 的密钥迁移协议提供了新的思路,我们下一步的工作是对 TPM 以及 vTPM 的迁移进行研究和分析,以此完善 TPM.

参考文献

[1] 冯登国,秦宇,汪丹,等. 可信计算技术研究[J]. 计算机研究与发展,2011,48(8):1332 – 1349.
FENG Deng-guo, QIN Yu, WANG Dan, et al. Research on trusted computing technology[J]. Journal of Computer Research and Development, 2011, 48(8): 1332 – 1349. (in Chinese)

[2] YU F, ZHANG H, ZHAO B, et al. A formal analysis of trusted platform module2.0 hash-based message authentication code authorization under digital rights management scenario[J]. Security & Communication Networks, 2016, 9(15): 2802 – 2815.

[3] 张焕国,韩文报,来学嘉,等. 网络空间安全综述[J]. 中国科学:信息科学,2016,46(2):125 – 164.
ZHANG Huan-guo, HAN Wen-bao, LAI Xue-jia, et al. Cyberspace security review[J]. Scientia Sinica Informationis, 2016, 46(2): 125 – 164. (in Chinese)

[4] 威尔·亚瑟,大卫·查林纳. TPM 2.0 原理及应用指南新安全时代的可信平台模块[M]. 北京:机械工业出版社,2017.
Will ARTHUR, David CHALLENER. A Practical Guide to TPM 2.0: Using the Trusted Platform Module in the New Age of Security[M]. Beijing: Machinery Industry Press, 2017. (in Chinese)

[5] 谭良,陈菊. 一种可信终端运行环境远程证明方案[J]. 软件学报,2014,25(6):1273 – 1290.
TAN Liang, CHEN Ju. Remote attestation project of the running environment of the trusted terminal[J]. Journal of Software, 2014, 25(6): 1273 – 1290. (in Chinese)

[6] 谭良,陈菊,周明天. 可信终端动态运行环境的可信证据收集机制[J]. 电子学报,2013,41(1):77 – 85.
TAN Liang, CHEN Ju. Unambiguous general framework design and applications for BOC signals[J]. Acta Electronica Sinica,2013,41(1):77 – 85. (in Chinese)

[7] 徐明迪,张焕国,张帆,等. 可信系统信任链研究综述[J]. 电子学报,2014,42(10):2024 – 2031.
XU Ming-di, ZHANG Huan-guo, ZHANG Fan, et al. Survey on chain of trust of trusted system[J]. Acta Electronica Sinica,2014,42(10):2024 – 2031. (in Chinese)

[8] 谭良,刘震,周明天. TCG 架构下的证明问题研究及进展[J]. 电子学报,2010,38(5):1105 – 1112.
TAN Liang, LIU Zhen, ZHOU Ming-tian. Development of attestation in TCG[J]. Acta Electronica Sinica, 2010, 38(5):1105 – 1112. (in Chinese)

[9] 徐甫. 基于可信根的计算机终端免疫模型[J]. 电子学报,2016,44(3):653 – 657.
XU Fu. Trusted root based computer terminal immune model[J]. Acta Electronica Sinica, 2016, 44(3):653 – 657. (in Chinese)

[10] 吴呈邑,熊焰,黄文超,等. 移动自组网中基于动态第三方的可信公平非抵赖协议[J]. 电子学报,2013,41(2):227 – 232.
WU Cheng-yi, XIONG Yan, HUANG Wen-chao, et al. A trusted fair non-repudiation protocol based on dynamic third party in mobile ad hoc networks[J]. Acta Electronica Sinica,2013,41(2):227 – 232. (in Chinese)

[11] 胡玲碧,谭良. 云计算中可信虚拟平台的远程证明方案研究[J/OL]. 软件学报, https://doi.org/10.13328/j.cnki.jos.005264,2018 – 06 – 08.
HU Ling-bi, TAN Liang. Research on the trusted virtual platform remote attestation method in cloud computing[J/OL]. Journal of Software, https://doi.org/10.13328/j.cnki.jos.005264,2018 – 06 – 08.

[12] 冯登国. 可信计算——理论与实践[M]. 北京:清华大学出版社,2013.
FENG Deng-guo. Trusted Computing Theory and Practice[M]. Beijing: Tsinghua University Press, 2013. (in Chinese)

[13] 陈军. 可信平台模块安全性分析与应用[D]. 北京:中国科学院计算技术研究所,2006.
CHEN Jun. Security Analysis of Trusted Platform Module and Application [D]. Beijing: Institute of Computing Technology Chinese Academy of Sciences, 2006. (in Chinese)

[14] 徐士伟,张焕国. 基于应用 π 演算的可信平台模块的安全性形式化分析[J]. 计算机研究与发展,2011,48(8):1421 – 1429.
XU Shi-wei, ZHANG Huan-guo. Formal security analysis on trusted platform module based on applied π calculus [J]. Journal of Computer Research and Development, 2011,48(8):1421 – 1429. (in Chinese)

[15] 张倩颖,赵世军,冯登国. TPM 可迁移密钥安全性分析与研究[J]. 小型微型计算机系统,2012,33(10):2188 – 2193.
ZHANG Qian-ying, ZHAO Shi-jun, FENG Deng-guo. Security analysis and research on TPM migratable key[J]. Journal of Chinese Computer Systems, 2012, 33(10): 2188 – 2193. (in Chinese)

[16] SHAO J, FENG D, QIN Y. Type-based analysis of protected storage in the TPM[A]. Information and Communications Security[M]. Berlin: Springer International Publishing, 2013. 135 – 150.

[17] ZHANG QY, ZHAO SJ, QIN Y, FENG DG. Formal analysis of TPM2.0 key management APIs[J]. Chinese Science Bulletin, 2014, 59:4210 – 4224.

[18] 徐扬,赵波,米兰·黑娜亚提,等. TPM2.0 密钥复制安全性增强方案[J]. 武汉大学学报(理学版),2014,60(6):471 – 477.
XU Yang, ZHAO Bo, MILAN Heinayati. Security enhancement of key duplication in TPM2.0[J]. Journal of Wuhan University (Natural Science Edition), 2014, 60(6):471 – 477. (in Chinese)

[19] 张倩颖,冯登国,赵世军. TCM 密钥迁移协议设计及形式化分析[J]. 软件学报,2015,26(9):2396 – 2417.
ZHANG Qian-ying, FENG Deng-guo, ZHAO Shi-jun. Design and formal analysis of TCM key migration protocols [J]. Journal of Software, 2015, 26(9):2396 – 2417. (in Chinese)

作者简介

宋 敏 女. 1994 年 6 月生,四川宜宾人. 现于四川师范大学计算机科学学院攻读硕士. 主要研究方向:可信计算,信息安全.
E-mail:songxiao8@yeah.net

谭 良 男. 1972 年 3 月生,四川泸州人,博士,教授. 现为四川师范大学继续教育学院副院长. 研究方向为可信计算、网络安全.