软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2015,26(9):2396−2417 [doi: 10.13328/j.cnki.jos.004719]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

TCM 密钥迁移协议设计及形式化分析*

张倩颖[1,2], 冯登国[1], 赵世军[1]

[1](中国科学院 软件研究所 可信计算与信息保障实验室,北京 100190)

[2](首都师范大学 信息工程学院,北京 100048)

通讯作者: 张倩颖, E-mail: zsjzqy@gmail.com

摘 要: 为增强 TCM 芯片间密钥的互操作性,TCM 提供了密钥迁移相关命令接口,允许用户设计密钥迁移协议以实现芯片间密钥的共享.通常,TCM 密钥迁移协议以目标 TCM 上的新父密钥作为迁移保护密钥.研究发现,该协议存在两个问题: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,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,以源 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.

关键词: 可信计算;可信密码模块;密钥迁移;协议设计;形式化分析

中图法分类号: TP309

中文引用格式: 张倩颖,冯登国,赵世军.TCM 密钥迁移协议设计及形式化分析.软件学报,2015,26(9):2396−2417. http://www.jos.org.cn/1000-9825/4719.htm

英文引用格式: Zhang QY, Feng DG, Zhao SJ. Design and formal analysis of TCM key migration protocols. Ruan Jian Xue Bao/ Journal of Software, 2015,26(9):2396−2417 (in Chinese). http://www.jos.org.cn/1000-9825/4719.htm

Design and Formal Analysis of TCM Key Migration Protocols

ZHANG Qian-Ying[1,2], FENG Deng-Guo[1], ZHAO Shi-Jun[1]

[1](Trusted Computing and Information Assurance Laboratory, Institute of Software, The Chinese Academy of Sciences, Beijing 100190, China)

[2](College of Information Engineering, Capital Normal University, Beijing 100048, China)

Abstract: TCM provides key migration interfaces to enhance interoperability between different TCM chips, allowing users to share keys between TCMs by key migration protocols. This study finds that the conventional TCM key migration protocol, which uses the new parent key of the migrated key on the destination TCM as the migration protection key, has two weaknesses. First, keys cannot be migrated to symmetric keys, which violates the design principles of TCM. Second, the absence of authentication between the originating TCM and destination TCM allows attacker to recover the migrated key of the originating TCM and to import his key into the destination TCM. To solve these issues, the paper proposes two new TCM key migration protocols. The first protocol, compliant with the TCM specification, allows keys to be migrated to symmetric keys and provides authentication of the destination TCM. The second protocol, which requires a slight modification to TCM key migration interfaces, not only solves all the two weaknesses, but also provides prefect forward security. Finally, the study formally analyzes the two protocols and demonstrates that the proposed protocols satisfy the correctness and desired security properties.

Key words: trusted computing; trusted cryptography module; key migration; protocol design; formal analysis

* 基金项目: 国家自然科学基金(91118006, 61202414); 国家重点基础研究发展计划(973)(2013CB338003)
收稿时间: 2014-01-23; 修改时间: 2014-07-16; 定稿时间: 2014-09-16

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].作为计算平台的信任基础,可信安全芯片提供了安全的存储功能、密码学计算功能,以及构建可信计算平台和远程证明所需的功能.目前,常用的可信安全芯片有两类:国际广泛使用的是符合可信平台模块(trusted platform module,简称 TPM)标准[2,3]的 TPM 芯片,我国普遍使用的是符合国家密码管理局发布的可信密码模块(trusted cryptography module,简称 TCM)标准[4]的 TCM 芯片.

TCM 是我国借鉴了国际可信计算技术框架与技术理念,在可信计算领域自主研制的安全芯片,其设计原则和技术思路与 TPM 存在较大不同:(1) TCM 以椭圆曲线密码算法和对称密码算法[5−7]为基础,计算效率高于 TPM;(2) TCM 密钥存储保护体系允许使用对称密钥作为存储密钥,提高了密钥管理的计算效率;(3) TCM 身份密钥符合双证书体制,更适用于我国的公钥基础设施[8];(4) TCM 提供了一些 TPM 不具备的功能,如数据对称加密和密钥协商功能;(5) 参考 TPM 已经暴露出的安全隐患[9],TCM 对自身的功能进行了改进,如 TCM 授权协议增加了对重放攻击的抵抗能力.

TCM 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.其中,密钥管理是 TCM 的一个重要功能,因为密钥的安全存储和使用是 TCM 能够有效地提供其他各项功能的前提和基础.TCM 用存储保护体系保证其密钥的安全性,同时,为满足用户在不同平台上使用同一密钥的需求,TCM 提供了密钥迁移功能,使用户能够在不同 TCM 之间迁移密钥.TCM 接口规范[10]详细规定了用于密钥迁移功能的命令接口,用户可以依据该类接口设计和实现不同的密钥迁移协议.

现有 TCM 密钥迁移协议通常以被迁移密钥在目标 TCM 上的新父密钥作为迁移保护密钥,用该密钥的公钥加密被迁移密钥实现迁移.本文通过分析发现,该协议存在两个问题:(1) 限制了被迁移密钥的新父密钥为非对称密钥,导致被迁移密钥不能迁移到目标 TCM 的对称存储密钥下,违背了 TCM 允许使用对称密钥作为父密钥的设计原则,不能充分利用 TCM 的高效性;(2) 作为协议参与方的源 TCM 和目标 TCM 未进行相互认证,导致被迁移密钥能够被敌手获得,并且敌手能够在 TCM 中嵌入自己的密钥,破坏了 TCM 存储保护体系的安全性.

针对上述问题,本文提出两个新的 TCM 密钥迁移协议:协议 1 在遵循 TCM 接口规范的情况下,以目标 TCM 的平台加密密钥作为迁移保护密钥,使源 TCM 能够在认证目标 TCM 身份之后,将密钥迁移至目标 TCM 任意类型的存储密钥下,保证了源 TCM 迁出密钥的安全性;考虑到现有 TCM 密钥迁移类接口无法实现源 TCM 身份认证的实际情况,协议 2 在简单修改 TCM 接口的情况下,以源 TCM 和目标 TCM 进行 SM2 密钥协商得到的会话密钥保护迁移数据,既不需要限制新父密钥的类型,又在协议交互过程中增加了参与双方 TCM 的相互认证,有效解决了上文提出的两个问题.此外,协议 2 还具有前向安全性,能够保证即使 TCM 用于 SM2 密钥协商的长期私钥泄露,也不会破坏已迁移密钥的安全性.

本文对上述两个协议进行了形式化分析:首先,用多集重写系统[11]对协议进行建模;然后,给出协议安全属性的形式化描述,包括协议正确性、源 TCM 迁出密钥的机密性、目标 TCM 迁入密钥的机密性以及协议 2 的前向安全性;最后,用自动定理证明工具 Tamarin[12]对上述安全属性进行验证.安全性分析结果表明:两个协议均满足预期的安全属性.

本文第 1 节简介 TCM 密钥迁移的相关背景知识.第 2 节介绍现有的 TCM 密钥迁移协议并指出其存在的问题.第 3 节详述本文提出的两个新的 TCM 密钥迁移协议.第 4 节对上述两个协议进行形式化分析,验证协议的安全性.第 5 节介绍可信安全芯片密钥迁移的相关研究工作.第 6 节总结全文.

1 背景知识

本节从 TCM 的密钥类型和结构、存储保护体系、密钥迁移类接口和密钥协商功能这 4 个方面介绍本文的背景知识.

1.1 密钥类型和结构

1.1.1 TCM 密钥类型

TCM 密钥按用途分为 6 种类型,本文将会用到其中的 3 种类型,即:

- 密码模块密钥(endorsement key,简称 EK):用于标识 TCM 及其所在平台的身份的一个 SM2 密钥.每个 TCM 芯片都有唯一的 *EK*,*EK* 私钥永久存储在芯片内部,只在少数关键操作中执行解密操作;
- 存储密钥(storage key,简称 SK):该密钥用于加密保护 TCM 存储保护体系中的密钥.按照所采用的加密算法,*SK* 分为采用 SM2 算法的非对称存储密钥和采用 SMS4 算法的对称存储密钥;
- 平台加密密钥(platform encryption key,简称 PEK):TCM 为我国的双证书体制引入的新类型密钥,是具有身份标识作用的 SM2 密钥.*PEK* 由可信第三方(trusted third party,简称 TTP)生成,同时生成的还有 *PEK* 的公钥证书.TCM 获得 *PEK* 以及 *PEK* 证书的流程是:TTP 使用 *EK* 加密的数字信封将 *PEK* 密钥以及 *PEK* 证书发送给 TCM,TCM 用 *EK* 私钥解密数字信封以获得 *PEK* 和 *PEK* 证书.

TCM 密钥按照是否具有迁移属性(migratable)可以分为可迁移密钥和不可迁移密钥两类:可迁移密钥可以从当前 TCM 迁移到另一个 TCM 中使用;不可迁移密钥只能在生成该密钥的 TCM 中使用.TCM 通过其内部安全机制保证用户只能对可迁移密钥实施迁移.

1.1.2 TCM 密钥结构

每个 TCM 密钥都分为公开区域和秘密区域两部分:前者包含指明密钥属性的各种字段,后者包含需要 TCM 保护的敏感字段.下面简介 TCM 密钥结构中与本文相关的一些字段.

- 公开区域
  1) *keyUsage*:密钥用途,该字段为 *TCM_SM2KEY_STORAGE* 或 *TCM_SMS4KEY_STORAGE* 的密钥是 *SK* 类型密钥,该字段为 *TCM_SM2KEY_PEK* 的密钥是 *PEK* 类型密钥;
  2) *pubKey*:密钥的公钥,对称密钥该字段值为空.
- 秘密区域
  1) *pubDataDigest*:密钥公开区域的摘要值;
  2) *key*:密钥的机密数据,对于非对称密钥,*key* 为其私钥,对于对称密钥,*key* 为密钥本身.

对任意密钥 *k*,本文用 *k*=(*keyPub*,*keyPri*)表示其结构,其中,*keyPub*=(*keyUsage*,*pubKey*)为其公开区域,*keyPri*=(*pubDataDigest*,*key*)为其秘密区域.

1.2 存储保护体系

TCM 芯片内部的存储空间非常有限,因此,将所有密钥都存储在 TCM 内部是不现实的.为解决密钥的安全存储问题,TCM 采用存储保护体系进行密钥管理,其思想是:利用密码学手段有效拓展 TCM 的存储空间,将密钥加密安全存储在 TCM 外部.

TCM 存储保护体系是一个树形结构,该结构以存储主密钥(storage master key,简称 SMK)为根节点,普通的存储密钥为中间节点,受保护的密钥或数据为叶节点.*SMK* 是 TCM 所有者获得 TCM 所有权时 TCM 产生的一个 SMS4 密钥,该密钥是特殊的存储密钥,仅在 TCM 内部存储和使用.除 *SMK* 之外,任何 TCM 密钥生成时,都必须指定存储保护体系中已存在的存储密钥作为其父密钥.当新生成的密钥需要存储到 TCM 外部时,TCM 用其父密钥对其秘密区域加密生成密钥包,将密钥包和公开区域存储在 TCM 外部.值得注意的是:TCM 允许使用对称密钥作为存储密钥保护子密钥,提高了密钥管理的效率,这是 TCM 有别于 TPM 的一大优势,而且这种设计原则也被下一代的 TPM 规范[3]所采纳.

1.3 密钥迁移接口

密钥迁移是指将一个 TCM(源 TCM)生成的密钥安全地转移至另一个 TCM(目标 TCM)的存储保护体系中,该功能主要用于密钥复制和备份.TCM 接口规范为用户提供以下 3 个密钥迁移类命令.本文用

$$Command(inputparams)\rightarrow(outputparams)$$

表示 TCM 命令,其中,*Command* 为命令名,*inputparams* 和 *outputparams* 分别为该命令的输入和输出参数列表.

命令 1. 保护密钥授权:*TCM_AuthorizeMigrationKey*(*migMode*,*mpkPub*)→(*migAuth*).

此命令以迁移模式 *migMode* 和迁移保护密钥公钥 *mpkPub* 为输入,计算迁移授权数据 *migAuth*,表明 TCM 所有者同意使用该迁移保护密钥(migration protection key,简称 MPK)在 *migMode* 模式下进行迁移.计算得出的 *migAuth* 包括以下内容:(1) *mpkPub*;(2) *migMode*;(3) *mpkPub*,*migMode* 和 *tcmProof* 三者的摘要值.其中,*tcmProof* 是 TCM 内部一个机密的随机数,只有 TCM 知道,即使 TCM 所有者也不能得到此值.

命令 2. 创建迁移数据:*TCM_CreateMigratedBlob*(hd_{PK},*migAuth*,*keyBlob*)→(*sblob*,*ablob*).

此命令使用迁移保护密钥公钥对被迁移密钥的秘密区域进行保护,生成被迁移密钥的迁移包(*sblob*,*ablob*).保护机制由迁移模式决定:若迁移模式为 *REWRAP*,则直接用迁移保护密钥公钥 *migAuth.mpkPub* 加密被迁移密钥的秘密区域得到 *ablob*,此模式下,*sblob* 为空;若迁移模式为 *MIGRATE*,则 TCM 首先生成对称密钥 *sk*,然后使用 *sk* 加密被迁移密钥的秘密区域得到 *sblob*,之后用迁移保护密钥公钥 *migAuth.mpkPub* 加密 *sk*,得到 *ablob*.

命令 3. 转换迁移数据:*TCM_ConvertMigratedBlob*(hd_{MPK},hd_{NPK},*sblob*,*ablob*)→(*keyBlob*).

TCM 首先获得被迁移密钥的秘密区域:如果 *sblob* 为空,表明迁移模式为 *REWRAP*,直接使用 hd_{MPK} 指向的密钥对 *ablob* 解密,即可获得被迁移密钥的秘密区域;如果 *sblob* 不为空,表明迁移模式为 *MIGRATE*,使用 hd_{MPK} 指向的密钥对 *ablob* 解密,得到对称密钥 *sk*,然后使用 *sk* 解密 *sblob*,获得被迁移密钥的秘密区域.之后,TCM 用新父密钥(new parent key,简称 NPK),即 hd_{NPK} 指向的密钥加密获得的秘密区域,得到一个可以载入到目标 TCM 存储保护体系中新父密钥下的密钥包.

利用上述命令实施密钥迁移的流程如图 1 所示.

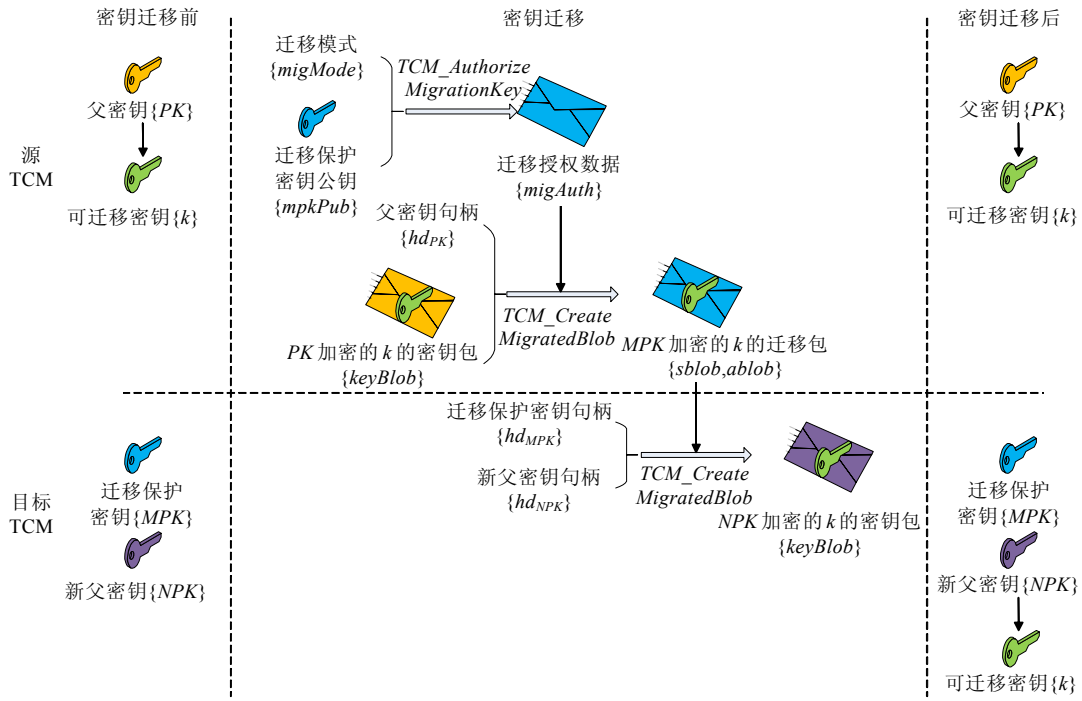

Fig.1 Process of TCM key migration

图 1 TCM 密钥迁移流程

在密钥迁移前,可迁移密钥 *k* 在源 TCM 的存储保护体系中,受其父密钥 *PK* 的保护,而迁移保护密钥 *MPK* 和新父密钥 *NPK* 则位于目标 TCM 的存储保护体系中.TCM 密钥迁移主要包括 3 个步骤:(1) 源 TCM 所有者利

用命令 1 对目标 TCM 的迁移保护密钥公钥 *mpkPub* 授权生成迁移授权数据,从而允许源 TCM 用户以该密钥为迁移保护密钥将可迁移密钥迁移到目标 TCM 中;(2) 源 TCM 用户以步骤(1)生成的迁移授权数据为参数调用命令 2,创建迁移数据,生成用 *MPK* 加密的 *k* 的迁移包;(3) 目标 TCM 用户以步骤(2)生成的迁移包为参数调用命令 3,将该迁移包转换为用 *NPK* 加密的 *k* 的密钥包.此后,可迁移密钥 *k* 就被迁移到目标 TCM 的存储保护体系中,受新父密钥 *NPK* 的保护.

1.4 密钥协商功能

利用 TCM 密钥协商功能,两个计算平台可以通过执行 SM2 密钥协商协议[5]建立共享的会话密钥,并保证该密钥始终处于芯片的保护之中.TCM 密钥协商包括 3 个步骤:交互双方的 TCM 各自生成临时 SM2 密钥;交互双方利用对方的公钥和临时密钥公钥、自身的私钥和临时密钥私钥,计算出共享的会话密钥;交互双方的 TCM 删除生成的临时密钥.由于共享密钥由交互双方的长期密钥和临时密钥共同计算得出,因此,该密钥协商协议提供了隐式的身份认证,交互双方可以确信只有对方可以得到协商出的共享密钥.

TCM 提供了 3 个密钥协商类命令,本文需要用到其中的两个.

命令 4. 创建协商会话:$TCM\_CreateKeyExchange()\rightarrow(X,shd_X)$.

此命令创建密钥协商会话,生成参与方的临时密钥$(x,X=g^x)$,并将其存储在会话中.其输入参数为空,输出参数为参与方的临时密钥公钥 X 和该密钥协商会话的句柄 shd_X.

命令 5. 释放协商会话:$TCM\_ReleaseExchangeSession(shd_X)\rightarrow()$.

此命令以密钥协商会话句柄 shd_X 为输入,用于释放该句柄指向的 TCM 密钥协商会话.

2 问题描述

TCM 密钥迁移协议包括 5 个参与实体:源 TCM、源 TCM 的所有者、目标 TCM、源 TCM 所在的主机、目标 TCM 所在的主机,其中,前 3 个实体是可信的,而源和目标 TCM 所在的主机被认为是不可信的.

本节首先介绍现在通用的 TCM 密钥迁移协议,然后详细分析该协议存在的问题.表 1 为本文使用的符号及函数说明.

Table 1 Symbols/Functions description

表 1 符号及函数描述

符号/函数	描述
T_A,O_A,H_A	源 TCM、源 TCM 所有者、源 TCM 所在的主机
T_B,O_B,H_B	目标 TCM、目标 TCM 所有者、目标 TCM 所在的主机
M,K_M,K_M^-	敌手、敌手公钥、敌手私钥
migMode,*migAuth*	迁移模式、*MPK* 的迁移授权数据
EK_N	某个 TCM T_N 的 *EK* 公钥
$PEK_N,PEK_N^-,PEKCert_N$	*PEK* 公钥、*PEK* 私钥、T_N 的 *PEK* 证书
PK_N,PK_N^-	被迁移密钥在 T_N 上的父密钥的公钥、私钥,如果父密钥为对称密钥则 $PK_N=PK_N^-$
keyPub,*keyPri*	被迁移密钥的公开区域、秘密区域
$keyBlob_N$	密钥包,即用 PK_N 加密的 *keyPri*
hd_k,shd_Y	密钥 *k* 的句柄、临时公钥为 *Y* 的密钥协商会话的句柄
null	空值
hash(*m*)	*m* 的哈希值
hmac(*m*,*k*)	用密钥 *k* 计算 *m* 的基于散列的消息认证码
senc(*m*,*k*),*sdec*(*m*,*k*)	用对称密钥 *k* 加密、解密 *m*
aenc(*m*,*k*),*adec*(*m*,*k*)	用公钥 *k* 加密 *m*、用私钥 *k* 解密 *m*
kdf(*f*,*s*)	以 *f* 为标志 *s* 为种子的密钥导出函数
$SM2KE(K_1,K_2,X,Y)$	用发起方的公钥 K_1 和临时密钥公钥 *X* 以及响应方的公钥 K_2 和临时密钥公钥 *Y*,计算 SM2 密钥协商协议的共享会话密钥

2.1 现在通用的密钥迁移协议

与 TPM 密钥迁移协议类似,现有 TCM 密钥迁移协议以被迁移密钥在目标 TCM 上的新父密钥(简称新父密钥)作为 *MPK*,其流程描述如下,协议流程图如图 2 所示.

1. $H_B \rightarrow H_A : PK_B$
2. O_A :调用$TCM_AuthorizeMigrationKey(migMode, PK_B)$
   T_A :计算$migAuth = (PK_B, migMode, hash(PK_B, migMode, tcmProof))$
3. $T_A \rightarrow H_A : migAuth$
4. H_A :调用$TCM_CreateMigratedBlob(hd_{PK_A}, migAuth, keyBlob_A)$
   T_A :1) 验证$migAuth$, 用hd_{PK_A}指向的PK_A^-解密$keyBlob_A$得到$keyPri$
   2) 根据$migMode$用PK_B保护$keyPri$:
   $REWRAP$: $sblob = null, ablob = aenc(keyPri, PK_B)$
   $MIGRATE$:生成对称密钥k, $sblob = senc(keyPri, k)$,
   $ablob = aenc(k, PK_B)$
5. $T_A \rightarrow H_A : sblob, ablob$
6. $H_A \rightarrow H_B : sblob, ablob, keyPub$
7. H_B :调用$TCM_ConvertMigratedBlob(hd_{PK_B}, hd_{PK_B}, sblob, ablob)$
   T_B :1) 根据$sblob$用第1个参数hd_{PK_B}指向的PK_B^-解密$keyPri$
   if $sblob = null : keyPri = adec(ablob, PK_B^-)$
   else : $k = adec(ablob, PK_B^-), keyPri = sdec(sblob, k)$
   2) 用第2个参数hd_{PK_B}指向的PK_B保护$keyPri$
   $keyBlob_B = aenc(keyPri, PK_B)$
8. $T_B \rightarrow H_B : keyBlob_B$

Fig.2 General key migration protocol for TCM

图 2 现在通用的 TCM 密钥迁移协议

(1) 传输保护密钥

H_B 将 PK_B 发送给 H_A.

(2) 保护密钥授权

1) O_A 调用 *TCM_AuthorizeMigrationKey*(*migMode*,PK_B);

2) T_A 接到调用后,计算 *migAuth*=(*PK*$_B$,*migMode*,*hash*(*PK*$_B$,*migMode*,*tcmProof*)),返回 *migAuth* 给 H_A.

(3) 创建迁移数据

1) H_A 调用 *TCM_CreateMigratedBlob*(hd_{PK_A},*migAuth*,$keyBlob_A$);

2) T_A 接到调用后,首先验证 *migAuth*,然后用 hd_{PK_A} 指向的 PK_A^- 解密 $keyBlob_A$ 得到 *keyPri*,之后,根据 *migMode* 用 PK_B 保护 *keyPri*:

- ✓ 若 *migMode=REWRAP*,则计算 *sblob=null*,*ablob=aenc*(*keyPri*,PK_B);
- ✓ 若 *migMode=MIGRATE*,则生成对称密钥 *k*,计算 *sblob=senc*(*keyPri*,*k*),*ablob=aenc*(*k*,PK_B);

最后返回 *sblob* 和 *ablob* 给 H_A.

(4) 传输迁移数据

H_A 将 *sblob*,*ablob* 和 *keyPub* 发送给 H_B.

(5) 转换迁移数据

1) H_B 调用 *TCM_ConvertMigratedBlob*(hd_{PK_B},hd_{PK_B},*sblob*,*ablob*);

2) T_B 接到调用后,首先根据 *sblob* 用第 1 个参数 hd_{PK_B} 指向的 PK_B^- 解密 *keyPri*:

- ✓ 若 *sblob=null*,则计算 $keyPri = adec(ablob, PK_B^-)$;
- ✓ 否则,计算 $k = adec(ablob, PK_B^-), keyPri = sdec(sblob, k)$;

然后,用第 2 个参数 hd_{PK_B} 指向的 PK_B 保护 *keyPri*,即,计算 $keyBlob_B$=*aenc*(*keyPri*,PK_B);

最后返回 $keyBlob_B$ 给 H_B 即完成了协议.

协议执行成功后,以 *keyPub*,$keyBlob_B$ 和新父密钥句柄 hd_{PK_B} 为参数,就可以将被迁移密钥加载到 T_B 的存储保护体系中.

2.2 存在的问题

- 问题 1

文献[13]指出,上述协议存在一个问题:由于 *SMK* 为对称密钥,故其无法作为被迁移密钥的新父密钥,所以被迁移密钥只能迁移到目标 TCM 存储保护体系的第 3 层或者更低层次.事实上,通过分析发现:该问题不仅只是被迁移密钥在目标 TCM 存储保护体系的层次问题,其实质是该协议以新父密钥作为 *MPK*,而 *MPK* 必须是非对称密钥,因此就限制了新父密钥只能为非对称密钥,从而导致被迁移密钥不能迁移到目标 TCM 的对称密钥下.而 TCM 相比于 TPM 的一个优势设计原则是允许使用对称密钥作为存储密钥以提高计算效率,所以此协议没有利用到 TCM 的优势,有悖于 TCM 设计初衷.

- 问题 2

我们研究发现,该协议缺少源 TCM 和目标 TCM 间的身份认证,导致密钥能够在敌手和 TCM 间迁移,即,图 3 中的两种情况:

1) 源 TCM 不能认证 *MPK* 是否是目标 TCM 的密钥,导致敌手可以用其控制的密钥迁移源 TCM 的密钥并获得密钥明文;

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

1. $H_B \to M : PK_B$
2. $M \to H_A : K_M$
3. O_A :调用$TCM_AuthorizeMigrationKey(K_M, migMode)$
   T_A :计算$migAuth = (K_M, migMode, hash(K_M, migMode, tcmProof))$
4. $T_A \to H_A : migAuth$
5. H_A :调用$TCM_CreateMigratedBlob(hd_{PK_A}, migAuth, keyBlob_A)$
   T_A :1) 验证$migAuth$,用hd_{PK_A}指向的PK_A^-解密$keyBlob_A$得到$keyPri$
   2) 根据$migMode$用K_M保护$keyPri$:
   $REWRAP : sblob = null, ablob = aenc(keyPri, K_M)$
   $MIGRATE$:生成对称密钥k, $sblob = senc(keyPri, k)$, $ablob = aenc(k, K_M)$
6. $T_A \to H_A : sblob, ablob$
7. $H_A \to M : sblob, ablob, keyPub$
8. M :根据$sblob$用其私钥K_M^-解密即得到$keyPri$
   if $sblob = null : keyPri = adec(ablob, K_M^-)$
   else : $k = adec(ablob, K_M^-), keyPri = sdec(sblob, k)$

(1)

1. $H_B \to M : PK_B$
2. M :1) 生成对称密钥或非对称密钥对
   2) 用生成的密钥根据TCM密钥结构构造$keyPub_M, keyPri_M$
   3) 任意选择$migMode$用PK_B保护$keyPri_M$:
   $REWRAP : sblob = null, ablob = aenc(keyPri_M, PK_B)$
   $MIGRATE$:生成对称密钥k, $sblob = senc(keyPri_M, k)$, $ablob = aenc(k, PK_B)$
3. $M \to H_B : sblob, ablob, keyPub_M$
4. H_B :调用$TCM_ConvertMigratedBlob(hd_{PK_B}, hd_{PK_B}, sblob, ablob)$
   T_B :1) 根据$sblob$用第一个参数hd_{PK_B}指向的PK_B^-解密$keyPri_M$
   if $sblob = null : keyPri_M = adec(ablob, PK_B^-)$
   else : $k = adec(ablob, PK_B^-), keyPri_M = sdec(sblob, k)$
   2) 用第二个参数hd_{PK_B}指向的PK_B保护$keyPri_M$
   $mkeyBlob_B = aenc(keyPri_M, PK_B)$
5. $T_B \to H_B : mkeyBlob_B$

(2)

Fig.3 Attacks on existing TCM key migration protocol

图 3 对现有 TCM 密钥迁移协议的攻击

为解决以上问题,本文提出两个新的 TCM 密钥迁移协议,在保证被迁移密钥可以以任意类型的存储密钥为新父密钥的基础上,最大限度的增强协议参与方间的认证性.值得一提的是:目前所有的可信安全芯片,如 TPMv1.2,TCM,TPM2.0,都没有能支持以对称存储密钥为新父密钥的密钥迁移协议.

3 新的 TCM 密钥迁移协议

本节介绍本文提出的两个 TCM 密钥迁移协议.

3.1 协议1

协议 1 使用目标 TCM 的 *PEK* 作为 *MPK* 进行迁移.此协议不需要改变 TCM 的命令接口,只需要增加新的迁移模式就能够实现,我们称该迁移模式为 *PEKMIGRATE*.协议 1 由 6 个阶段组成:初始化、传输保护密钥、保护密钥授权、创建迁移数据、传输迁移数据和转换迁移数据,其流程描述如下,协议流程图如图 4 所示.

(1) 初始化

1) T_B 将 *PEK* 参数发送给 TTP;

2) TTP 首先按照给定参数生成 PEK_B, PEK_B^- 和 $PEKCert_B$,然后用 PEK_B 和 PEK_B^- 根据 TCM 密钥结构构造 $key_{PEK_B}=(keyPub_{PEK_B},keyPri_{PEK_B})$,之后生成对称密钥 k_1 和 k_2,计算:

$$a=senc(key_{PEK_B},k_1),b=aenc(k_1,EK_B),c=senc(PEKCert_B,k_2),d=aenc(k_2,EK_B);$$

最后,将 *a*,*b*,*c* 和 *d* 发送给 T_B;

3) T_B 打开 *a*,*b* 和 *c*,*d* 两个数字信封,得到 key_{PEK_B} 和 $PEKCert_B$.

(2) 传输保护密钥

H_B 将 PEK_B 和 $PEKCert_B$ 发送给 H_A.

(3) 保护密钥授权

1) O_A 验证 PEK_B 和 $PEKCert_B$ 后,调用 *TCM_AuthorizeMigrationKey*(*migMode*,PEK_B),其中,

migMode=*PEKMIGRATE*;

2) T_A 接到调用后,计算 *migAuth*=(PEK_B,*migMode*,*hash*(PEK_B,*migMode*,*tcmProof*)),返回 *migAuth* 给 H_A.

(4) 创建迁移数据

1) H_A 调用 *TCM_CreateMigratedBlob*(hd_{PK_A},*migAuth*,$keyBlob_A$);

2) T_A 接到调用后,首先验证 *migAuth*,然后用 hd_{PK_A} 指向的 PK_A^- 解密 $keyBlob_A$ 得到 *keyPri*,之后,根据 *migMode* 用 PEK_B 保护 *keyPri*:

a) 生成随机种子 *seed*,导出 k_1=*kdf*('*encryption*',*seed*),k_2=*kdf*('*integrity*',*seed*);

b) 计算 *a*=*senc*(*keyPri*,k_1),*b*=*hmac*(*a*,k_2),*sblob*=(*b*,*a*);

c) 计算 *ablob*=*aenc*(*seed*,PEK_B),

最后,返回 *sblob* 和 *ablob* 给 H_A.

(5) 传输迁移数据

H_A 将 *sblob*、*ablob* 和 *keyPub* 发送给 H_B.

(6) 转换迁移数据

1) H_B 调用 *TCM_ConvertMigratedBlob*(hd_{PEK_B},hd_{PK_B},*sblob*,*ablob*);

2) T_B 接到调用后,首先用 hd_{PEK_B} 指向的 PEK_B^- 恢复出 *keyPri*:

a) 计算 $seed=adec(ablob,PEK_B^-)$,导出 k_1=*kdf*('*encryption*',*seed*),k_2=*kdf*('*integrity*',*seed*);

b) 将 *sblob* 的前 32 字节赋值给 *b*,余下字节赋值给 *a*;

c) 验证 *b*=*hmac*(*a*,k_2)后,计算 *keyPri*=*sdec*(*a*,k_1);

然后,用 hd_{PK_B} 指向的 PK_B 保护 *keyPri*:

✓ 若 PK_B 为非对称密钥,则计算 $keyBlob_B$=*aenc*(*keyPri*,PK_B);

✓ 若 PK_B 为对称密钥,则计算 $keyBlob_B$=*senc*(*keyPri*,PK_B),

最后,返回 $keyBlob_B$ 给 H_B 即完成了协议.

此协议在数字信封的基础上加入了 *HMAC* 校验,用于保证迁移数据的完整性.目标 TCM 在接收到迁移数据后,首先检查 *HMAC* 校验值:如果不能通过校验则不进行迁移数据转换,省去了后续的解密步骤.

首先,此协议的一个优势是新父密钥可以是任意类型的存储密钥,即,允许使用高效率的对称密钥作为新父

密钥;此外,此协议使用 PEK_B 作为 MPK,O_A 通过验证 PEK_B 和 $PEKCert_B$ 可以确定其来自 T_B,增加了 T_A 对 MPK 来源的认证,能够有效防止图 3 中的第 1 种攻击.

1. $T_B \rightarrow TTP$: PEK参数
2. TTP :1) 生成PEK_B, PEK_B^-,构造TCM密钥结构$key_{PEK_B} = (keyPub_{PEK_B}, keyPri_{PEK_B})$
   2) 生成对称密钥k_1,计算$a = senc(key_{PEK_B}, k_1), b = aenc(k_1, EK_B)$
   3) 为PEK_B签发证书$PEKCert_B$
   4) 生成对称密钥k_2,计算$c = senc(PEKCert_B, k_2), d = aenc(k_2, EK_B)$
3. $TTP \rightarrow T_B$: a, b, c, d
4. T_B :1) 计算$k_1 = adec(b, EK_B^-), key_{PEK_B} = sdec(a, k_1)$
   2) 计算$k_2 = adec(d, EK_B^-), PEKCert_B = sdec(c, k_2)$
5. $H_B \rightarrow H_A$: $PEK_B, PEKCert_B$
6. O_A :验证PEK_B和$PEKCert_B$,调用$TCM_AuthorizeMigrationKey(PEKMIGRATE, PEK_B)$
   T_A :计算$migAuth = (PEK_B, PEKMIGRATE, hash(PEK_B, PEKMIGRATE, tcmProof))$
7. $T_A \rightarrow H_A$: $migAuth$
8. H_A :调用$TCM_CreateMigratedBlob(hd_{PK_A}, migAuth, keyBlob_A)$
   T_A :1) 验证$migAuth$,用hd_{PK_A}指向的PK_A^-解密$keyBlob_A$得到$keyPri$
   2) 根据$migMode = PEKMIGRATE$用PEK_B保护$keyPri$:
   a) 生成随机种子$seed$,计算$k_1 = kdf('encryption', seed), k_2 = kdf('integrity', seed)$
   b) 计算$a = senc(keyPri, k_1), b = hmac(a, k_2), sblob = (a, b)$
   c) 计算$ablob = aenc(seed, PEK_B)$
9. $T_A \rightarrow H_A$: $sblob, ablob$
10. $H_A \rightarrow H_B$: $sblob, ablob, keyPub$
11. H_B :调用$TCM_ConvertMigratedBlob(hd_{PEK_B}, hd_{PK_B}, sblob, ablob)$
    T_B :1) 用hd_{PEK_B}指向的PEK_B^-恢复出$keyPri$
    a) 计算$seed = adec(ablob, PEK_B^-), k_1 = kdf('encryption', seed), k_2 = kdf('integrity', seed)$
    b) 将$sblob$前32字节赋值给b,其余字节赋值给a
    c) 验证$b = hmac(a, k_2)$,计算$keyPri = sdec(a, k_1)$
    2) 用hd_{PK_B}指向的PK_B保护$keyPri$
    if PK_B为非对称密钥: $keyBlob_B = aenc(keyPri, PK_B)$
    else: $keyBlob_B = senc(keyPri, PK_B)$
12. $T_B \rightarrow H_B$: $keyBlob_B$

Fig.4 First protocol

图 4 协议 1

3.2 协议2

通过分析 TCM 密钥迁移类命令接口,发现现有的 3 个接口均不能提供对源 TCM 身份的认证,导致目标 TCM 不能确定其收到的迁移数据是否真正来自源 TCM,因此也就无法防止图 3 中的第 2 种攻击.另外,协议 1 使用 PEK 保护被迁移密钥,而 PEK 是由外部的可信第三方生成后导入 TCM 内部的,如果 PEK 在外部泄露,那么所有被迁移密钥将不再安全,敌手可以获得被迁移密钥的秘密区域.因此,需要在密钥迁移协议中增加前向安全性(perfect forward secrecy,简称 PFS),以保证已经迁移过的密钥的安全性.

为解决上述问题,本文提出了协议 2.此协议使用 SM2 密钥协商协议得到的会话密钥保护被迁移密钥,迁移协议完成后,TCM 删除临时密钥,使敌手不能再解密迁移数据.协议 2 需要简单修改 TCM 密钥迁移类接口,我们称此协议的迁移模式为 $KEYEXCHANGE$.

3.2.1 接口修改

协议 2 需要修改第 1.3 节中的命令 2、命令 3,在此分别标记为命令 6、命令 7.

命令 6. 创建迁移数据:$TCM\_CreateMigratedBlob(hd_{PK}, migAuth, hd_{PEK_A}, Y, keyBlob_A) \rightarrow (X, sblob, ablob)$.

此命令使用源 TCM 和目标 TCM 进行 SM2 密钥协商协议得到的会话密钥对被迁移密钥的秘密区域进行保护,其保护机制与协议一的 *PEKMIGRATE* 模式相同.为计算与目标 TCM 共享的密钥,该命令比命令 2 中的原接口增加了两个输入参数:源 TCM *PEK* 密钥句柄 hd_{PEK_A}、目标 TCM 临时密钥公钥 Y 和一个输出参数:源 TCM 临时密钥公钥 X.

命令 7. 转换迁移数据:*TCM_ConvertMigratedBlob*(hd_{PEK_B},shd_Y,PEK_A,X,hd_{PK_B},*sblob*,*ablob*)→(*keyBlob*$_B$).

该命令首先使用目标 TCM 的 *PEK* 私钥、临时密钥私钥和源 TCM 的 *PEK* 公钥、临时密钥公钥计算 SM2 密钥协商协议的共享密钥,用该密钥恢复被迁移密钥的秘密区域;然后,用新父密钥对其进行保护,得到一个可以载入目标 TCM 存储保护体系中新父密钥下的密钥包.为计算与源 TCM 共享的密钥,该命令比命令 3 中的原接口增加了 3 个输入参数:目标 TCM 密钥协商会话句柄 shd_Y、源 TCM *PEK* 公钥 PEK_A、源 TCM 临时密钥公钥 X.

3.2.2 协议描述

协议 2 在原 TCM 密钥迁移协议 5 个参与实体的基础上,增加了一个参与实体:目标 TCM 的所有者,该实体被认为是可信的.协议 2 的流程描述如下,协议流程图如图 5 所示.

(1) 初始化

T_A 和 T_B 采用与协议 1 中 T_B 相同的初始化方式进行初始化.初始化完成后,T_A 得到 key_{PEK_A} 和 $PEKCert_A$,T_B 得到 key_{PEK_B} 和 $PEKCert_B$.

(2) 传输保护密钥

H_B 将 PEK_B 和 $PEKCert_B$ 发送给 H_A.

(3) 保护密钥授权

1) O_A 验证 PEK_B 和 $PEKCert_B$ 后,调用 *TCM_AuthorizeMigrationKey*(*migMode*,PEK_B),其中:

migMode=*KEYEXCHANGE*;

2) T_A 接到调用后,计算 *migAuth*=(*PEK*$_B$,*migMode*,*hash*(*PEK*$_B$,*migMode*,*tcmProof*)),返回 *migAuth* 给 H_A.

(4) 传输临时密钥

1) H_B 调用 *TCM_CreateKeyExchange*();

2) T_B 接到调用后,创建密钥协商会话,随机生成会话句柄 shd_Y 和临时密钥私钥 y,计算临时密钥公钥 $Y=g^y$,将临时密钥对(y,Y)存储在芯片内部与句柄 shd_Y 绑定,返回 Y,shd_Y 给 H_B;

3) H_B 将 Y 发送给 H_A.

(5) 创建迁移数据

1) H_A 调用 *TCM_CreateMigratedBlob*(hd_{PK_A},*migAuth*,hd_{PEK_A},Y,*keyBlob*$_A$);

2) T_A 接到调用后,首先验证 *migAuth*,然后用 hd_{PK_A} 指向的 PK_A^- 解密 *keyBlob*$_A$ 得到 *keyPri*,之后,根据 *migMode* 生成 *seed* 保护 *keyPri*:

a) 随机生成临时密钥私钥 x,计算临时密钥公钥 $X=g^x$;

b) 计算 *seed*=*SM2KE*(*PEK*$_B$,*PEK*$_A$,Y,X),导出 k_1=*kdf*('*encryption*',*seed*),k_2=*kdf*('*integrity*',*seed*);

c) 计算 *a*=*senc*(*keyPri*,k_1),*b*=*hmac*(*a*,k_2),*sblob*=(*b*,*a*),*ablob*=*null*;

最后,返回 X,*sblob* 和 *ablob* 给 H_A,并删除临时密钥对(x,X).

(6) 传输迁移数据

H_A 将 PEK_A,$PEKCert_A$,X,*sblob*,*ablob* 和 *keyPub* 发送给 H_B.

(7) 转换迁移数据

1) O_B 验证 PEK_A 和 $PEKCert_A$ 后,调用:

TCM_ConvertMigratedBlob(hd_{PEK_B},shd_Y,PEK_A,X,hd_{PK_B},*sblob*,*ablob*);

2) T_B接到调用后,首先用hd_{PEK_B}指向的PEK_B^-、shd_Y指向的y,PEK_A,X计算$seed$,并恢复出$keyPri$:

a) 计算$seed=SM2KE(PEK_B,PEK_A,Y,X)$,导出$k_1=kdf(\text{'}encryption\text{'},seed)$,$k_2=kdf(\text{'}integrity\text{'},seed)$;

b) 将$sblob$的前32字节赋值给b,余下字节赋值给a;

c) 验证$b=hmac(a,k_2)$后,计算$keyPri=sdec(a,k_1)$;

然后,用hd_{PK_B}指向的PK_B保护$keyPri$:

✓ 若PK_B为非对称密钥,则计算$keyBlob_B=aenc(keyPri,PK_B)$;

✓ 若PK_B为对称密钥,则计算$keyBlob_B=senc(keyPri,PK_B)$;

最后,返回$keyBlob_B$给H_B.

(8) 删除临时密钥

1) O_B调用$TCM_ReleaseExchangeSession(shd_Y)$;

2) T_B接到调用后,删除shd_Y指向的临时密钥对(y,Y)即完成了协议.

1. $T_A \rightarrow TTP$: PEK参数
2. TTP :1) 生成PEK_A,PEK_A^-, 构造TCM密钥结构$key_{PEK_A}=(keyPub_{PEK_A},keyPri_{PEK_A})$
   2) 生成对称密钥k_1,计算$a=senc(key_{PEK_A},k_1),b=aenc(k_1,EK_A)$
   3) 为PEK_A签发证书$PEKCert_A$
   4) 生成对称密钥k_2,计算$c=senc(PEKCert_A,k_2),d=aenc(k_2,EK_A)$
3. $TTP \rightarrow T_A$: a,b,c,d
4. T_A :计算$k_1=adec(b,EK_A^-),key_{PEK_A}=sdec(a,k_1),k_2=adec(d,EK_A^-),PEKCert_A=sdec(c,k_2)$
5. $T_B \rightarrow TTP$: PEK参数
6. TTP :1) 生成PEK_B,PEK_B^-, 构造TCM密钥结构$key_{PEK_B}=(keyPub_{PEK_B},keyPri_{PEK_B})$
   2) 生成对称密钥k_1,计算$a=senc(key_{PEK_B},k_1),b=aenc(k_1,EK_B)$
   3) 为PEK_B签发证书$PEKCert_B$
   4) 生成对称密钥k_2,计算$c=senc(PEKCert_B,k_2),d=aenc(k_2,EK_B)$
7. $TTP \rightarrow T_B$: a,b,c,d
8. T_B :计算$k_1=adec(b,EK_B^-),key_{PEK_B}=sdec(a,k_1),k_2=adec(d,EK_B^-),PEKCert_B=sdec(c,k_2)$
9. $H_B \rightarrow H_A$: $PEK_B,PEKCert_B$
10. O_A :验证PEK_B和$PEKCert_B$,调用$TCM_AuthorizeMigrationKey(KEYEXCHANGE,PEK_B)$
    T_A :计算$migAuth=(PEK_B,KEYEXCHANGE,hash(PEK_B,KEYEXCHANGE,tcmProof))$
11. $T_A \rightarrow H_A$: $migAuth$
12. H_B :调用$TCM_CreateKeyExchange()$
    T_B :1) 创建密钥协商会话,随机生成会话句柄shd_Y
    2) 随机生成临时密钥私钥y,计算临时密钥公钥$Y=g^y$
    3) 将临时密钥对(y,Y)存储在TCM中并与句柄shd_Y绑定
13. $T_B \rightarrow H_B$: Y,shd_Y
14. $H_B \rightarrow H_A$: Y
15. H_A :调用$TCM_CreateMigratedBlob(hd_{PK_A},migAuth,hd_{PEK_A},Y,keyBlob_A)$
    T_A :1) 验证$migAuth$,用hd_{PK_A}指向的PK_A^-解密$keyBlob_A$得到$keyPri$
    2) 根据$migMode=KEYEXCHANGE$生成$seed$保护$keyPri$:
    a) 随机生成临时密钥私钥x,计算临时密钥公钥$X=g^x$
    b) 计算$seed=SM2KE(PEK_B,PEK_A,Y,X),k_1=kdf(\text{'}encryption\text{'},seed),k_2=kdf(\text{'}integrity\text{'},seed)$
    c) 计算$a=senc(keyPri,k_1),b=hmac(a,k_2),sblob=(a,b),ablob=null$
    d) 删除临时密钥私钥x
16. $T_A \rightarrow H_A$: $X,sblob,ablob$
17. $H_A \rightarrow H_B$: $PEK_A,PEKCert_A,X,sblob,ablob,keyPub$

Fig.5 Second protocol

图5 协议2

18. O_B :验证PEK_A和$PEKCert_A$,调用$TCM_ConvertMigratedBlob(hd_{PEK_B}, shd_Y, PEK_A, X, hd_{PK_B}, sblob, ablob)$

T_B :1) 用hd_{PEK_B}指向的PEK_B^-,shd_Y指向的y,PEK_A,X计算$seed$并恢复出$keyPri$

a) 计算$seed = SM2KE(PEK_B, PEK_A, Y, X), k_1 = kdf('encryption', seed), k_2 = kdf('integrity', seed)$

b) 将$sblob$前32字节赋值给b,其余字节赋值给a

c) 验证$b = hmac(a, k_2)$,计算$keyPri = sdec(a, k_1)$

2) 用hd_{PK_B}指向的PK_B保护$keyPri$

if PK_B为非对称密钥: $keyBlob_B = aenc(keyPri, PK_B)$

else: $keyBlob_B = senc(keyPri, PK_B)$

19. $T_B \rightarrow H_B$:$keyBlob_B$

20. O_B :调用$TCM_ReleaseExchangeSession(shd_Y)$

T_B :删除shd_Y指向的临时密钥对(y, Y)

Fig.5 Second protocol (continued)

图 5 协议 2 (续)

首先,该协议不需要用新父密钥公钥保护迁移过程,从而免去了对新父密钥类型的限制,允许被迁移密钥迁移到目标 TCM 任意类型的存储密钥下;其次,此协议使用 T_A 和 T_B 的 *PEK* 作为长期密钥进行密钥协商,O_A 和 O_B 通过验证 *PEK* 和 *PEK* 证书可以认证 *PEK* 的来源,从而实现了交互双方 TCM 的相互认证,能够防止图 3 中的两种攻击;最后,此协议在迁移完成后删除参与计算 SM2 会话密钥的临时密钥,保证以后即使双方 *PEK* 私钥泄漏,敌手也不能计算出 SM2 会话密钥获得被迁移的密钥,从而保证了密钥迁移协议的前向安全性.

3.3 协议比较

对现有 TCM 密钥迁移协议(P0)和本文提出的协议 1(P1)、协议 2(P2)的比较结果见表 2.可以看出:

- 在符合 TCM 接口规范的情况下,P1 比 P0 具有更好的适用性和安全性,即:既允许 *NPK* 为对称存储密钥,又能实现对目标 TCM 的身份认证;
- 而在修改 TCM 接口的情况下,P2 比 P1 具有更强的安全性,该协议不仅增加了对源 TCM 身份的认证,而且满足了前向安全性.

Table 2 Comparisons of the three key migration protocols

表 2 3 个密钥迁移协议比较

项	P0	P1	P2
非对称 *NPK*	√	√	√
对称 *NPK*	×	√	√
目标 TCM 身份认证	×	√	√
源 TCM 身份认证	×	×	√
前向安全性	×	×	√
符合 TCM 接口规范	√	√	×

4 形式化分析

首先介绍本文所采用的形式化分析方法,然后详细说明对协议 1 和协议 2 的正确性和安全性的证明过程.

4.1 分析方法介绍

本文用多集重写系统(multiset rewriting system)建模 TCM 密钥迁移协议,将协议步骤和敌手行为形式化描述为多集重写规则,并将协议的安全属性描述为一阶逻辑公式.然后,将上述模型编码为自动定理证明工具 Tamarin 的脚本,使用 Tamarin 对密钥迁移协议进行形式化分析.本节简介多集重写系统的语法、语义以及属性描述语言和证明工具 Tamarin,详细介绍参见文献[11].

4.1.1 基本语法

定义 1. 项(term)用于建模密码学消息,由以下语法定义:

$$\begin{array}{rcll} t,t_i & ::= & x & x\in V \\ & | & n & n\in FN\cup PN \\ & | & f(t_1,...,t_n) & f\in \Sigma_T \end{array}.$$

其中,V 是变量集合;FN 是新鲜值集合;PN 是公开值集合;Σ_T 是以下函数符号的集合:

$$\Sigma_T=\{aenc(\_,\_),\ senc(\_,\_),\ sign(\_,\_),\ hash(\_,\_),\ smke(\_,\_,\_,\_),\ \langle\_,\_\rangle,$$
$$adec(\_,\_),\ sdec(\_,\_),\ verify(\_,\_,\_),\ hmac(\_,\_),\ pk(\_),\ kdf(\_),\ true,\ null\}.$$

其中,$\langle_,_\rangle$表示二元组,后文将$\langle\langle a,b\rangle,c\rangle$和$\langle a,\langle b,c\rangle\rangle$简写为$\langle a,b,c\rangle$;函数 $smke$ 用于计算 SM2 密钥协商协议的共享秘密.

我们将Σ_T中函数符号的代数性质形式化为等价理论 E_T,E_T包含以下等式:

$$E_T=\left\{\begin{array}{l} adec(aenc(m,pk(k)),k)\simeq m,\quad verify(sign(m,k),m,pk(k))\simeq true, \\ sdec(senc(m,k),k)\simeq m,\quad smke(a,pk(b),x,pk(y))\simeq smke(pk(a),b,pk(x),y) \end{array}\right\}.$$

定义 2. 事实(fact)定义为 $f:=F(t_1,\ldots,t_n),t_i\in T,F\in\Sigma_F$.其中:$T$ 是项集合;Σ_F 是以下事实符号的集合:

$$\Sigma_F=\{K(\_),\ Out(\_),\ In(\_),\ Fr(\_),\ !Cert(\_,\_),\ !TTPKpair(\_,\_),\ RequestPEK(\_),$$
$$SHandle(\_,\_,\_,\_),\ !Handle(\_,\_,\_,\_,\_),\ !TCMEK(\_,\_),\ !TCMProof(\_,\_)\}.$$

事实符号分为线性的和持久性的:前者建模只能使用一次的资源;后者除 $K(_)$外均以"!"为前缀,用于建模可以任意使用的无穷资源.第 1 行符号分别建模敌手知识、输出消息、输入消息、新鲜值信息、实体公钥证书、TTP 公私钥对和 TCM 的 *PEK* 请求;第 2 行符号分别建模 TCM 密钥协商会话句柄、TCM 密钥句柄、TCM 的 *EK* 公钥和 TCM 的 *tcmProof*.例如:

- $!Handle(A,hd,keyUsage,pubKey,key)$是一个持久性事实,表明 TCM A 中存在一个密钥句柄,该句柄值为 hd,指向一个用途为 $keyUsage$ 的密钥,其公钥为 $pubKey$,私钥为 key;
- $SHandle(A,shd,pubKey,key)$表明 TCM A 中存在一个密钥协商会话句柄,该句柄值为 shd,绑定的临时密钥对为$(pubKey,key)$.该事实是一个线性事实,可以通过执行 *TCM_ReleaseExchangeSession* 命令被消耗.

本文用 G 表示不包含变量的事实的集合.

定义 3. 一个多集重写规则由前提 l、动作 a 和结论 r 组成,表示为 $l-[a]\to r$,其中,l,a 和 r 是事实序列.对于多集重写规则集合 R,$ginst(R)$表示 R 中不包含变量的实例的集合.

定义 4. 消息推理规则 MD 由以下规则组成:

$$MD=\{Out(x)-[]\to K(x),\quad K(x)-[K(x)]\to In(x),\quad Fr(x:fresh)-[]\to K(x:fresh),$$
$$[]-[]\to K(x:pub),\quad K(x_1),...,K(x_k)-[]\to K(f(x_1,...,x_k))\mid f\in\Sigma_T\}.$$

其中,$x:s$ 表示 x 是类型为 s 的变量.第 1 行规则分别建模敌手能够从公开信道接收消息、向公开信道发送消息和获得公开值;第 2 行规则分别建模敌手能够获得自己产生的新鲜值和对自己已知的消息应用Σ_T中的函数.

定义 5. 规则 $FRESH=([]-[]\to Fr(x:fresh))$表示生成新鲜值.所有新鲜值都由 $FRESH$ 规则的唯一实例生成,即,该规则总是产生不同的新鲜值.

4.1.2 操作语义

多集重写规则的语义由一个转换系统(transition system)给出,转换系统的状态 S 是事实的多集.

记法:对于集合 A,$P(A)$表示 A 的幂集,$A^{\#}$表示 A 中元素的有限多集组成的集合.对于运算符,上标"#"表示多集上的运算,如$\cup^{\#}$表示两个多集的并运算,$\Phi^{\#}$表示空多集.

定义 6. 对于多集重写系统 R 和等价理论 E_T,转换关系 $\to_{R,E_T}\subseteq G^{\#}\times P(G)\times G^{\#}$ 由以下转换规则定义:

$$\frac{l-[a]\to r\in_{E_T} ginsts(R\cup\{FRESH\})\quad lfacts(l)\subseteq^{\#} S\quad pfacts(l)\subseteq set(S)}{S\xrightarrow{set(a)}_{R,E_T}((S\setminus^{\#} lfacts(l))\cup^{\#} mset(r))}.$$

其中,$lfact(l)$是 l 中线性事实的多集,$pfact(l)$是 l 中持久性事实的集合,set 表示多集或序列所对应的集合,$mset$ 表

示序列所对应的多集,$\in_{E_T}$ 表示模 E_T 的属于关系.

该转换规则建模一个重写规则实例重写状态 S 的过程:首先,$\subseteq^{\#}$检查 $lfact(l)$中的线性事实在 S 中发生足够多次,$\subseteq$检查 $pfact(l)$中的持久性事实都在 S 中发生;然后,通过去掉 S 中已消耗的线性事实并增加新产生的事实,得到 S 的后继状态.转换关联的标记 $set(a)$是规则实例对应的动作集合.

定义 7. 迹(trace)集合用于建模协议执行,其定义为

$$traces_{E_T}(R) := \{[A_1,...,A_n] \mid \exists S_1,...,S_n \in G^{\#}.\quad \Phi^{\#} \xrightarrow{A_1}_{R,E_T} S_1 \xrightarrow{A_2}_{R,E_T} ... \xrightarrow{A_n}_{R,E_T} S_n \wedge$$
$$\forall i \neq j.\quad \forall x.\quad (S_{i+1} \setminus^{\#} S_i) = \{Fr(x)\}^{\#} \Rightarrow (S_{j+1} \setminus^{\#} S_j) \neq \{Fr(x)\}^{\#}\}.$$

其中,第 2 个条件确保迹中 *FRESH* 规则的每个实例都不同.

4.1.3 属性描述语言

协议的安全属性建模为转换系统的迹属性,由一阶逻辑公式描述.

定义 8. 迹公式(trace formula)是由迹原子(trace atom)产生的一阶公式.迹原子有以下 5 种类型:1) 恒假$\bot$;2) 项相等 $t_1 \approx t_2$;3) 时间点排序 $i<j$;4) 时间点相等 $i \doteq j$;5) 表示事实 f 在时间点 i 发生的动作 $f@i$.

定义 9. 对于等价理论 E_T,迹 tr、赋值θ和迹公式φ间的可满足关系 $(tr,\theta) \vDash_{E_T} \varphi$ 定义为:

$(tr,\theta) \vDash_{E_T} \bot$ never

$(tr,\theta) \vDash_{E_T} f@i$ iff $\theta(i) \in idx(tr)$ and $f\theta \in_{E_T} tr_{\theta(i)}$

$(tr,\theta) \vDash_{E_T} i < j$ iff $\theta(i) < \theta(j)$

$(tr,\theta) \vDash_{E_T} i \doteq j$ iff $\theta(i) = \theta(j)$

$(tr,\theta) \vDash_{E_T} t_1 \approx t_2$ iff $t_1\theta =_{E_T} t_2\theta$

$(tr,\theta) \vDash_{E_T} \neg\varphi$ iff not $(tr,\theta) \vDash_{E_T} \varphi$

$(tr,\theta) \vDash_{E_T} \varphi \wedge \psi$ iff $(tr,\theta) \vDash_{E_T} \varphi$ and $(tr,\theta) \vDash_{E_T} \psi$

$(tr,\theta) \vDash_{E_T} \exists x{:}s.\ \varphi$ iff there is $u \in \boldsymbol{D}_s$ such that $(tr,\theta[x \mapsto u]) \vDash_{E_T} \varphi$

其中,$idx(tr)$表示 tr 索引的集合,$tr_{\theta(i)}$表示 tr 的第$\theta(i)$个元素,$\boldsymbol{D}_s$ 表示类型 s 的定义域,$\theta[x \mapsto u]$表示将 x 映射到 u、将 $a(a \neq x)$映射到$\theta(a)$的函数.

定义 10. φ是 R,E 可满足的(R,E-satisfiable),记为 $R \vDash^{\exists}_{E_T} \varphi$,当且仅当存在迹 $tr \in traces_{E_T}(R)$ 和赋值θ,使得 $(tr,\theta) \vDash_{E_T} \varphi$ 成立.

定义 11. φ是 R,E 有效的(R,E-valid),记为 $R \vDash^{\forall}_{E_T} \varphi$,当且仅当 $(tr,\theta) \vDash_{E_T} \varphi$ 对任意 $tr \in traces_{E_T}(R)$ 和任意赋值θ都成立.

4.1.4 证明工具 Tamarin

Tamarin[12]是一个符号化的验证工具,支持安全协议的证伪和验证.该工具以描述为多集重写系统的协议作为输入,将其翻译为依赖图,然后用约束求解算法来验证表示为一阶逻辑公式的协议安全属性.对于可满足性断言 $R \vDash^{\exists}_{E_T} \varphi$,Tamarin 验证协议迹集合中是否存在满足$\varphi$的迹;对于有效性断言 $R \vDash^{\forall}_{E_T} \varphi$,Tamarin 验证协议迹集合中是否存在不满足$\varphi$的迹.Tamarin 不仅能验证 Dolev-Yao 模型下的安全属性,而且能验证复杂模型下的安全属性,如密钥协商协议中 eCK 模型下的安全属性[14].

4.2 协议1分析

本节在 Dolev-Yao 模型下证明协议 1 的正确性和安全性.

4.2.1 协议描述

协议 1 中不可信实体可以被敌手控制,其行为和敌手行为一样,都通过第 4.1 节中的规则集合 *MD* 描述.协议 1 中可信实体的形式化描述如图 6 所示,我们称该规则集合为 P1.

1. Init for TTP:
   $[Fr(tk)]-[OneTime()]\rightarrow[Out(pk(tk)),!TTPKpair(tk,pk(tk)),!Cert(TTP,pk(tk))]$
2. Issue for TTP:
   $[RequestPEK(A),!TCMEK(A,pk(ek)),!TTPKpair(tk,pk(tk)),Fr(pek),Fr(k_1),Fr(k_2)]-[]\rightarrow[Out(\langle a,b,c,d\rangle)]$
   where $keyPub=\langle'TCM\_SM2KEY\_PEK',pk(pek)\rangle, keyPri=\langle hash(keyPub),pek\rangle,$
   $a=senc(\langle keyPub,keyPri\rangle,k_1), b=aenc(a,pk(ek)), c=senc(sign(\langle A,pk(pek)\rangle,tk),k_2), d=aenc(c,pk(ek))$
3. Activate and load PEK for T:
   $[In(\langle a,b,c,d,ekh,smkh\rangle),!Handle(A,ekh,'TCM\_SM2KEY\_EK',pk(ek),ek),!Handle(A,smkh,'TCM\_SMS4KEY\_STORAGE',null,smk),$
   $Fr(pekh)]-[]\rightarrow[Out(\langle pekh,keyPub,keyBlob,sign(\langle A,pk(pek)\rangle,tk)\rangle),!Handle(A,pekh,'TCM\_SM2KEY\_PEK',pk(pek),pek)]$
   where $keyPub=\langle'TCM\_SM2KEY\_PEK',pk(pek)\rangle, keyPri=\langle hash(keyPub),pek\rangle, a=senc(\langle keyPub,keyPri\rangle,k_1), b=aenc(a,pk(ek)),$
   $c=senc(sign(\langle A,pk(pek)\rangle,tk),k_2), d=aenc(c,pk(ek)), keyBlob=senc(keyPri,smk)$
4. LoadKey to SMS4 parent key for T:
   $[In(snpkh,keyPub,keyBlob),!Handle(A,snpkh,'TCM\_SMS4KEY\_STORAGE',null,snpk),Fr(kh)]$
   $-[TCMLoadKey(A,'SMS4',k,snpk)]\rightarrow[!Handle(A,kh,'TCM\_SM2KEY\_STORAGE',pk(k),k),Out(kh)]$
   where $keyPub=\langle'TCM\_SM2KEY\_STORAGE',pk(k)\rangle, keyPri=\langle hash(keyPub),k\rangle, keyBlob=senc(keyPri,snpk)$
5. LoadKey to SM2 parent key for T:
   $[In(anpkh,keyPub,keyBlob),!Handle(A,anpkh,'TCM\_SM2KEY\_STORAGE',pk(anpk),anpk),Fr(kh)]$
   $-[TCMLoadKey(A,'SM2',k,anpk)]\rightarrow[!Handle(A,kh,'TCM\_SM2KEY\_STORAGE',pk(k),k),Out(kh)]$
   where $keyPub=\langle'TCM\_SM2KEY\_STORAGE',pk(k)\rangle, keyPri=\langle hash(keyPub),k\rangle, keyBlob=aenc(keyPri,pk(anpk))$
6. Init for T_A:
   $[Fr(tcmProof),Fr(opk),Fr(opkh),Fr(k)]-[]\rightarrow$
   $[Out(\langle keyPub,keyBlob,opkh\rangle),!TCMProof(A,tcmProof),!Handle(A,opkh,'TCM\_SM2KEY\_STORAGE',pk(opk),opk)]$
   where $keyPub=\langle'TCM\_SM2KEY\_STORAGE',pk(k)\rangle, keyPri=\langle hash(keyPub),k\rangle, keyBlob=aenc(keyPri,pk(opk))$
7. AuthorizeMigrationKey for T_A:
   $[In(migMode,pk(pek),cert),!Cert(TTP,pk(tk)),!TCMProof(A,tcmProof)]-[Eq(verify(cert,\langle B,pk(pek)\rangle,pk(tk)),true)]\rightarrow[Out(migAuth)]$
   where $cert=sign(\langle B,pk(pek)\rangle,tk), migAuth=\langle migMode,pk(pek),hash(migMode,pk(pek),tcmProof)\rangle$
8. CreateMigratedBlob for T_A:
   $[In(opkh,migAuth,keyBlob),!TCMProof(A,tcmProof),!Handle(A,opkh,'TCM\_SM2KEY\_STORAGE',pk(opk),opk),Fr(seed)]$
   $-[TCMCreateMigratedBlob(A,B,k,pk(pek))]\rightarrow[Out(\langle sblob,ablob\rangle)]$
   where $migAuth=\langle'PEKMIGRATE',pk(pek),hash('PEKMIGRATE',pk(pek),tcmProof)\rangle, keyPub=\langle'TCM\_SM2KEY\_STORAGE',pk(k)\rangle,$
   $keyPri=\langle hash(keyPub),k\rangle, keyBlob=aenc(keyPri,pk(opk)), k_1=kdf('encryption',seed), k_2=kdf('integrity',seed),$
   $a=senc(keyPri,k_1), b=hmac(a,k_2), sblob=\langle a,b\rangle, ablob=aenc(seed,pk(pek))$
9. Init for T_B:
   $[Fr(ek),Fr(ekh),Fr(smk),Fr(smkh),Fr(snpk),Fr(snpkh),Fr(anpk),Fr(anpkh)]-[]\rightarrow[Out(\langle ekh,smkh,snpkh,anpkh\rangle),RequestPEK(B),$
   $!TCMEK(B,pk(ek)),!Handle(B,ekh,'TCM\_SM2KEY\_EK',pk(ek),ek),!Handle(B,smkh,'TCM\_SMS4KEY\_STORAGE',null,smk),$
   $!Handle(B,snpkh,'TCM\_SMS4KEY\_STORAGE',null,snpk),!Handle(B,anpkh,'TCM\_SM2KEY\_STORAGE',pk(anpk),anpk)]$
10. ConvertMigratedBlob to SMS4 for T_B:
    $[!Handle(B,pekh,'TCM\_SM2KEY\_PEK',pk(pek),pek),!Handle(B,snpkh,'TCM\_SMS4KEY\_STORAGE',null,snpk),$
    $In(pekh,snpkh,sblob,ablob)]-[TCMConvertMigratedBlob(B,'SMS4',k,snpk)]\rightarrow[Out(keyBlob)]$
    where $keyPub=\langle'TCM\_SM2KEY\_STORAGE',pk(k)\rangle, keyPri=\langle hash(keyPub),k\rangle, k_1=kdf('encryption',seed), k_2=kdf('integrity',seed),$
    $a=senc(keyPri,k_1), b=hmac(a,k_2), sblob=\langle a,b\rangle, ablob=aenc(seed,pk(pek)), keyBlob=senc(keyPri,snpk)$
11. ConvertMigratedBlob to SM2 for T_B:
    $[!Handle(B,pekh,'TCM\_SM2KEY\_PEK',pk(pek),pek),!Handle(B,anpkh,'TCM\_SM2KEY\_STORAGE',pk(anpk),anpk),$
    $In(pekh,anpkh,sblob,ablob)]-[TCMConvertMigratedBlob(B,'SM2',k,anpk)]\rightarrow[Out(keyBlob)]$
    where $keyPub=\langle'TCM\_SM2KEY\_STORAGE',pk(k)\rangle, keyPri=\langle hash(keyPub),k\rangle, k_1=kdf('encryption',seed), k_2=kdf('integrity',seed),$
    $a=senc(keyPri,k_1), b=hmac(a,k_2), sblob=\langle a,b\rangle, ablob=aenc(seed,pk(pek)), keyBlob=aenc(keyPri,pk(anpk))$

Fig.6 Multiset rewriting rules formalizing the first protocol

图 6 协议 1 的多集重写规则描述

规则 1、规则 2 建模可信第三方的行为.规则 1 描述 TTP 初始化,该规则生成 TTP 私钥,发送 TTP 公钥.事实$!TTPKpair(tk,pk(tk))$表示 TTP 的私钥和公钥分别为 tk 和 $pk(tk)$;事实$!Cert(TTP,pk(tk))$表示 TTP 持有 $pk(tk)$的公钥证书;动作 $OneTime()$表示 TTP 仅初始化 1 次,被用在安全属性中的迹公式α_{OT}:=$(\forall ij.OneTime()@i\wedge OneTime()@j\Rightarrow i=j)$中,该公式表示发生动作 $OneTime()$的时间点都是相同的,从而限制规则 1 只能实例化 1 次,

即,TTP 只能产生唯一的证书颁发密钥和对应的公钥证书.规则 2 描述 TTP 为 TCM 生成 *PEK* 和颁发 *PEK* 证书,该规则接收 TCM *A* 的 *PEK* 请求,查找 TCM *A* 的 *EK* 公钥和 TTP 的公私钥对,然后为 TCM *A* 生成 *PEK* 和 *PEK* 证书,计算用 TCM *A* 的 *EK* 公钥加密的两个数字信封,两个数字信封分别包含生成的 *PEK* 密钥和 *PEK* 证书,并发送这两个数字信封.

规则 3~规则 5 建模源和目标 TCM 的行为,事实符号!*Handle* 中变量 *A* 的取值用以区分该规则实例发生在哪一个 TCM 中.规则 3 描述 TCM 激活和加载 *PEK*,该规则接收包含 *PEK* 密钥和 *PEK* 证书的两个数字信封、TCM 的 *EK* 密钥句柄和 *SMK* 密钥句柄,读取上述句柄对应的密钥信息,然后为 *PEK* 密钥生成句柄,发送该句柄、*PEK* 的公开区域、*SMK* 加密的 *PEK* 密钥包和 *PEK* 证书,并存储 *PEK* 密钥句柄对应的密钥信息.规则 4、规则 5 分别描述 TCM 用对称和非对称父密钥加载子密钥,这两个规则接收父密钥句柄、子密钥的公开区域和密钥包,读取父密钥句柄对应的密钥信息,然后为子密钥生成句柄,存储子密钥句柄对应的密钥信息,并发送该句柄.两个规则的区别在于父密钥句柄对应的密钥类型和子密钥密钥包的加密方式不同:规则 4 中为对称密钥和对称加密,规则 5 中为非对称密钥和非对称加密.规则 4 中动作 *TCMLoadKey*(*A*,'*SMS*4',*k*,*snpk*)表示 TCM *A* 以 SMS4 密钥 *snpk* 为父密钥加载了密钥 *k*;类似地,规则 5 中动作 *TCMLoadKey*(*A*,'*SM*4',*k*,*anpk*)表示 TCM *A* 以 SM2 密钥 *anpk* 为父密钥加载了密钥 *k*.

规则 6~规则 8 建模源 TCM 的行为.规则 6 描述源 TCM 初始化,该规则生成源 TCM 的 *tcmProof*、存储密钥、存储密钥句柄和可迁移密钥,发送可迁移密钥的公开区域、存储密钥加密的可迁移密钥的密钥包和存储密钥句柄,并存储 *tcmProof* 值和存储密钥句柄对应的密钥信息.规则 7 描述源 TCM 授权保护密钥,该规则接收迁移模式、目标 TCM 的 *PEK* 公钥和 *PEK* 证书,查找 TTP 的公钥证书和源 TCM 的 *tcmProof* 值,计算迁移授权数据,并发送该迁移授权数据.动作 *Eq*(*verify*(*cert*,⟨*B*,*pk*(*pek*)⟩,*pk*(*tk*)),*true*)描述源 TCM 所有者验证目标 TCM 的 *PEK* 和 *PEK* 证书,即,用 TTP 公钥 *pk*(*tk*)验证证书 *cert* 是对身份标识 *B* 和 *PEK* 公钥 *pk*(*pek*)的签名,该动作被用在安全属性中的迹公式α_{Eq}:=($\forall x\ y\ i.\quad Eq(x,y)@i \Rightarrow x=y$)中.$\alpha_{Eq}$过滤掉分析过程中包含 *Eq*(*x*,*y*)且 $x\neq y$ 的迹,从而保证只有 *PEK* 证书验证成功,源 TCM 所有者才会授权该 *PEK* 为迁移保护密钥.规则 8 描述源 TCM 创建迁移数据,该规则接收被迁移密钥的父密钥句柄、迁移授权数据和被迁移密钥的密钥包,查找源 TCM 的 *tcmProof* 值和被迁移密钥的父密钥句柄对应的密钥信息,生成随机种子,计算迁移包,并发送该迁移包.动作 *TCMCreateMigratedBlob*(*A*,*B*,*k*,*pk*(*pek*))表示 TCM *A* 以 TCM *B* 的 *pk*(*pek*)为迁移保护密钥创建了密钥 *k* 的迁移包.

规则 9~规则 11 建模目标 TCM 的行为.规则 9 描述目标 TCM 初始化,该规则生成目标 TCM 的 *EK*、*EK* 密钥句柄、*SMK*、*SMK* 密钥句柄、对称存储密钥、对称存储密钥句柄、非对称存储密钥和非对称存储密钥句柄,发送上述 4 个句柄,产生 *PEK* 请求,并存储 *EK* 公钥和 4 个密钥句柄对应的密钥信息.规则 10、规则 11 分别描述目标 TCM 将密钥迁移至对称和非对称密钥下,这两个规则接收目标 TCM 的 *PEK* 密钥句柄、被迁移密钥的新父密钥句柄和迁移包,查找上述两个句柄对应的密钥信息,计算新父密钥加密的被迁移密钥的密钥包,并发送该密钥包.两个规则的区别在于,新父密钥句柄对应的密钥类型和被迁移密钥密钥包的加密方式不同:规则 10 中是对称密钥和对称加密,规则 11 中是非对称密钥和非对称加密.规则 10 中动作 *TCMConvertMigratedBlob* (*B*,'*SMS*4',*k*,*snpk*)表示 TCM *B* 以 SMS4 密钥 *snpk* 为新父密钥转换了密钥 *k* 的迁移数据;类似地,规则 11 中动作 *TCMConvertMigratedBlob*(*B*,'*SM*2',*k*,*anpk*)表示 TCM *B* 以 SM2 密钥 *anpk* 为新父密钥转换了密钥 *k* 的迁移数据.

4.2.2 安全属性验证

定理 1(key migration correctness). $\mathrm{P1} \cup MD_{\Sigma_T} \models^{\exists}_{E_T} \alpha_{OT} \wedge \alpha_{Eq} \Rightarrow \varphi_1 \wedge \varphi_2$,其中,

$$\varphi_1 := \exists A\ B\ mk\ pek\ npk\ i\ j\ k.\quad TCMCreateMigratedBlob(A,B,mk,pk(pek))@i \wedge TCMConvertMigratedBlob(B,'SMS4',mk,npk)@j \wedge TCMLoadKey(B,'SMS4',mk,npk)@k \wedge A \neq B \wedge i \lessdot j \wedge j \lessdot k,$$

$$\varphi_2 := \exists A\ B\ mk\ pek\ npk\ i\ j\ k.\quad TCMCreateMigratedBlob(A,B,mk,pk(pek))@i \wedge TCMConvertMigratedBlob(B,'SM2',mk,npk)@j \wedge TCMLoadKey(B,'SM2',mk,npk)@k \wedge A \neq B \wedge i \lessdot j \wedge j \lessdot k.$$

定理 1 表示在假设α_{OT}和α_{Eq}成立的情况下,存在 P1 的执行满足φ_1和φ_2,即:在 TTP 仅初始化一次,并且只有目标 TCM 的 *PEK* 证书验证成功时,源 TCM 所有者才授权该 *PEK* 为迁移保护密钥的情况下,存在协议 1 的执行满足安全属性φ_1和φ_2.安全属性φ_1表示存在 TCM *A* 在时间点 *i* 以 TCM *B* 的 *pk*(*pek*)为迁移保护密钥创建密钥 *mk* 的迁移包,TCM *B* 在 *i* 之后的时间点 *j* 以 SMS4 密钥 *npk* 为新父密钥转换了 *mk* 的迁移数据,并在 *j* 之后的时间点 *k* 以 *npk* 为父密钥加载了 *mk*,且 TCM *A* 和 TCM *B* 为不同的 TCM.安全属性φ_2表示存在 TCM *A* 在时间点 *i* 以 TCM *B* 的 *pk*(*pek*)为迁移保护密钥创建密钥 *mk* 的迁移包,TCM *B* 在 *i* 之后的时间点 *j* 以 SM2 密钥 *npk* 为新父密钥转换了 *mk* 的迁移数据,并在 *j* 之后的时间点 *k* 以 *npk* 为父密钥加载了 *mk*,且 TCM *A* 和 TCM *B* 为不同的 TCM.因此,定理 1 表明:存在协议 1 的执行满足源 TCM 迁出的密钥能够被迁移到目标 TCM 的对称(φ_1)和非对称(φ_2)密钥下,并且目标 TCM 能够用新父密钥加载被迁移的密钥.该定理用于验证协议 1 的正确性.

定理 2(key migration security). $P1 \cup MD_{\Sigma_T} \models^{\forall}_{E_T} \alpha_{OT} \wedge \alpha_{Eq} \Rightarrow \varphi_3 \wedge \varphi_4$,其中,

$\varphi_3 := \forall A\ B\ mk\ pek\ i.\quad TCMCreateMigratedBlob(A,B,mk,pk(pek))@i \Rightarrow \neg(\exists j.\quad K(mk)@j)$,

$\varphi_4 := \forall A\ type\ mk\ npk\ i.\quad TCMConvertMigratedBlob(A,type,mk,npk)@i \Rightarrow \neg(\exists j.\quad K(mk)@j)$.

定理 2 表示在假设α_{OT}和α_{Eq}成立的情况下,P1 的所有执行都满足φ_3和φ_4.即:在 TTP 仅初始化 1 次,并且只有目标 TCM 的 *PEK* 证书验证成功时,源 TCM 所有者才授权该 *PEK* 为迁移保护密钥的情况下,协议 1 的所有执行都满足安全属性φ_3和φ_4.安全属性φ_3表示 TCM *A* 在时间点 *i* 以 TCM *B* 的 *pk*(*pek*)为迁移保护密钥创建密钥 *mk* 的迁移包,那么不存在时间点 *j*,在该时间点 *mk* 被敌手获得.安全属性φ_4表示 TCM *A* 在时间点 *i* 以任意类型的密钥 *npk* 为新父密钥转换了 *mk* 的迁移数据,那么不存在时间点 *j*,在该时间点 *mk* 被敌手获得.因此,定理 2 表明,协议 1 的所有执行都能够保证从源 TCM 迁出的密钥(φ_3)和迁移到目标 TCM 的密钥(φ_4)不会被敌手获得.该定理用于验证协议 1 能够保证源 TCM 迁出密钥和目标 TCM 迁入密钥的机密性,即,验证协议一能够保证密钥迁移的安全性.

以规则集合 P1 和定理 1、定理 2 的安全属性编码为脚本作为 Tamarin 的输入,经过 Tamarin 验证得出如图 7 所示的结果.图中第 1 行文字提示后续输出为 Tamarin 的验证结果,第 2 行文字表示验证的协议文件名为 Tcm_Key_Migration_Protocol1.spthy,之后的 4 行输出即是协议安全属性的验证结果,其格式为:序号_安全属性名称(安全属性类型):安全属性验证结果(验证执行的步骤数).安全属性有两种类型:exists-trace 表示该安全属性是可满足性断言,all-traces 表示该安全属性是有效性断言.验证结果同样有两种:verified 表示该安全属性验证为真,falsified-found trace 表示该安全属性验证为假,Tamarin 发现不满足该安全属性的迹.图中的 4 行输出分别对应定理 1、定理 2 中φ_1~φ_4的验证结果.与协议 1 预期的安全属性一致,φ_1~φ_3验证为真,φ_4验证为假.Tamarin 给出的对安全属性φ_4的攻击流程与图 3 中的第 2 种攻击相同,即:由于目标 TCM 不能认证迁移数据的来源,使敌手能将其已知的密钥迁移到目标 TCM 中.

```
==============================================================================
summary of summaries:

analyzed: Tcm_Key_Migration_Protocol1.spthy

  1_can_migrate_key_to_sms4parentkey (exists-trace): verified (14 steps)
  2_can_migrate_key_to_sm2parentkey (exists-trace): verified (14 steps)
  3_migratedkey_security (all-traces): verified (373 steps)
  4_importedkey_security (all-traces): falsified - found trace (14 steps)

==============================================================================
```

Fig.7 Analysis results of the first protocol by Tamarin

图 7 协议 1 的 Tamarin 验证结果

4.3 协议2分析

文献[15]给出了一个可证明安全性的 SM2 密钥协商协议,本节在此基础上证明协议 2 的安全属性.限于篇幅,本节将重点介绍第 4.2 节中未涉及的内容.而对于协议 2 与协议 1 在分析过程中的共同之处,只做简要说明.

4.3.1 协议描述

描述协议 2 的规则集合为 P2.P2 是以下两个集合的并集:图 6 中规则 1~规则 5 和规则 7 组成的集合;图 8

所示的规则集合.

1. Init for T_A :

$$[Fr(tcmProof'),Fr(ek),Fr(ekh),Fr(smk),Fr(smkh),Fr(opk),Fr(opkh),Fr(k)]-[]\rightarrow[Out(\langle ekh,smkh,opkh,keyPub,keyBlob\rangle),$$
$$!TCMProof(A,tcmProof'),RequestPEK(A),!TCMEK(A,pk(ek)),!Handle(A,ekh,'TCM\_SM2KEY\_EK',pk(ek),ek),$$
$$!Handle(A,smkh,'TCM\_SMS4KEY\_STORAGE',null,smk),!Handle(A,opkh,'TCM\_SM2KEY\_STORAGE',pk(opk),opk)]$$

where $keyPub=\langle'TCM\_SM2KEY\_STORAGE',pk(k)\rangle,keyPri=\langle hash(keyPub),k\rangle,keyBlob=aenc(keyPri,pk(opk))$

2. CreateMigratedBlob for T_A :

$$[In(opkh,migAuth,apekh,pk(y),keyBlob),!TCMProof(A,tcmProof'),!Handle(A,opkh,'TCM\_SM2KEY\_STORAGE',pk(opk),opk),$$
$$!Handle(A,apekh,'TCM\_SM2KEY\_PEK',pk(apek),apek),Fr(x)]$$
$$-[TCMCreateMigratedBlob(A,B,k,pk(bpek))]\rightarrow[Out(\langle pk(x),sblob,ablob\rangle)]$$

where $migAuth=\langle'KEYEXCHANGE',pk(bpek),hash('KEYEXCHANGE',pk(bpek),tcmProof')\rangle,keyPub=\langle'TCM\_SM2KEY\_STORAGE',pk(k)\rangle,$
$keyPri=\langle hash(keyPub),k\rangle,keyBlob=aenc(keyPri,pk(opk)),seed=smke(pk(bpek),apek,pk(y),x)$
$k_1=kdf('encryption',seed),k_2=kdf('integrity',seed),a=senc(keyPri,k_1),b=hmac(a,k_2),sblob=\langle a,b\rangle,ablob=null$

3. CreateKeyExchange for T_B :

$$[Fr(y),Fr(ykh)]-[]\rightarrow[Out(\langle pk(y),ykh\rangle),SHandle(B,ykh,pk(y),y)]$$

4. ConvertMigratedBlob to SMS4 for T_B :

$$[In(bpekh,ykh,pk(apek),pk(x),snpkh,sblob,ablob,cert),!Cert(TTP,pk(tk)),!Handle(B,bpekh,'TCM\_SM2KEY\_PEK',pk(bpek),bpek),$$
$$SHandle(B,ykh,pk(y),y),!Handle(B,snpkh,'TCM\_SMS4KEY\_STORAGE',null,snpk)]$$
$$-[Eq(verify(cert,\langle A,pk(apek)\rangle,pk(tk)),true),TCMConvertMigratedBlob(B,'SMS4',k,snpk,y)]\rightarrow[Out(keyBlob),SHandle(B,ykh,pk(y),y)]$$

where $cert=sign(\langle A,pk(apek)\rangle,tk),keyPub=\langle'TCM\_SM2KEY\_STORAGE',pk(k)\rangle,keyPri=\langle hash(keyPub),k\rangle,$
$seed=smke(bpek,pk(apek),y,pk(x)),k_1=kdf('encryption',seed),k_2=kdf('integrity',seed),$
$a=senc(keyPri,k_1),b=hmac(a,k_2),sblob=\langle a,b\rangle,ablob=null,keyBlob=senc(keyPri,snpk)$

5. ConvertMigratedBlob to SM2 for T_B :

$$[In(bpekh,ykh,pk(apek),pk(x),anpkh,sblob,ablob,cert),!Cert(TTP,pk(tk)),!Handle(B,bpekh,'TCM\_SM2KEY\_PEK',pk(bpek),bpek),$$
$$SHandle(B,ykh,pk(y),y),!Handle(B,anpkh,'TCM\_SM2KEY\_STORAGE',pk(anpk),anpk)]-$$
$$[Eq(verify(cert,\langle A,pk(apek)\rangle,pk(tk)),true),TCMConvertMigratedBlob(B,'SM2',k,anpk,y)]\rightarrow[Out(keyBlob),SHandle(B,ykh,pk(y),y)]$$

where $cert=sign(\langle A,pk(apek)\rangle,tk),keyPub=\langle'TCM\_SM2KEY\_STORAGE',pk(k)\rangle,keyPri=\langle hash(keyPub),k\rangle,$
$seed=smke(bpek,pk(apek),y,pk(x)),k_1=kdf('encryption',seed),k_2=kdf('integrity',seed),$
$a=senc(keyPri,k_1),b=hmac(a,k_2),sblob=\langle a,b\rangle,ablob=null,keyBlob=aenc(keyPri,pk(anpk))$

6. ReleaseExchangeSession for T_B :

$$[In(ykh),SHandle(B,ykh,pk(y),y)]-[TCMReleaseExchangeSession(y)]\rightarrow[Out('success')]$$

7. Longterm key reveal for M :

$$[In(sign(\langle A,pk(pek)\rangle,tk)),!Cert(TTP,pk(tk))]-[RevealLtk(A)]\rightarrow[Out(pek)]$$

Fig.8 Multiset rewriting rules formalizing the second protocol

图 8 协议 2 的多集重写规则描述

下面简介图 8 中各规则的语义.

规则 1、规则 2 建模源 TCM 的行为.规则 1 描述源 TCM 初始化,该规则生成源 TCM 的 *tcmProof*、*EK*、*EK* 密钥句柄、*SMK*、*SMK* 密钥句柄、存储密钥和存储密钥句柄,发送上述 3 个句柄,存储 *tcmProof* 值,产生 *PEK* 请求,并存储 *EK* 公钥和 3 个密钥句柄对应的密钥信息.规则 2 描述源 TCM 创建迁移数据,该规则接收被迁移密钥的父密钥句柄、迁移授权数据、源 TCM 的 *PEK* 密钥句柄、目标 TCM 的临时密钥公钥和被迁移密钥的密钥包,查找源 TCM 的 *tcmProof* 值和上述两个密钥句柄对应的密钥信息,生成源 TCM 的临时密钥私钥,计算迁移包,发送源 TCM 的临时密钥公钥和该迁移包.动作 *TCMCreateMigratedBlob*(*A*,*B*,*k*,*pk*(*bpek*))表示 TCM *A* 以 TCM *B* 的 *pk*(*pek*)为迁移保护密钥创建了密钥 *k* 的迁移包.

规则 3~规则 6 建模目标 TCM 的行为.规则 3 描述目标 TCM 创建协商会话,该规则生成目标 TCM 的临时密钥私钥和密钥协商会话句柄,发送目标 TCM 的临时密钥公钥和该密钥协商会话句柄,并将生成的临时密钥对存储在该密钥协商会话句柄对应的会话信息中.规则 4、规则 5 分别描述目标 TCM 迁移密钥至对称和非对称密钥下,这两个规则接收目标 TCM 的 *PEK* 密钥句柄、目标 TCM 的密钥协商会话句柄、源 TCM 的 *PEK* 公钥、源 TCM 的临时密钥公钥、被迁移密钥的新父密钥句柄、被迁移密钥的迁移包和源 TCM 的 *PEK* 证书,查找 TTP 的公钥证书和上述 3 个句柄对应的信息,计算新父密钥加密的被迁移密钥的密钥包,发送该密钥包,并重新存储

该密钥协商会话句柄对应的会话信息.与图 6 中的规则 7 类似,这两个规则中的动作 *Eq* 描述目标 TCM 所有者验证源 TCM 的 *PEK* 和 *PEK* 证书,被用在安全属性中的迹公式α_{Eq}中,从而保证只有 *PEK* 证书验证成功,目标 TCM 所有者才会允许迁移数据转换至目标 TCM 的存储保护体系中.两个规则的区别在于新父密钥句柄对应的密钥类型和被迁移密钥的密钥包加密方式不同:规则 4 中为对称密钥和对称加密,规则 5 中为非对称密钥和非对称加密.规则 4 中动作 *TCMConvertMigratedBlob*(*B*,'*SMS*4',*k*,*snpk*,*y*)表示 TCM *B* 以 SMS4 密钥 *snpk* 为新父密钥 *y* 为临时密钥私钥转换了密钥 *k* 的迁移数据;类似地,规则 5 中动作 *TCMConvertMigratedBlob*(*B*,'*SM*2',*k*,*anpk*,*y*)表示 TCM *B* 以 SM2 密钥 *anpk* 为新父密钥 *y* 为临时密钥私钥转换了密钥 *k* 的迁移数据.规则 6 描述目标 TCM 释放协商会话,该规则接收目标 TCM 的密钥协商会话句柄,查找该句柄对应的会话信息,释放该句柄指向的密钥协商会话,并发送表示密钥协商会话释放成功的信息.动作 *TCMReleaseExchangeSession*(*y*)表示目标 TCM 释放了临时密钥私钥 *y* 对应的密钥协商会话.

规则 7 建模敌手行为,该规则接收 TCM 的 *PEK* 证书,查找 TTP 的公钥证书,发送 *PEK* 私钥.动作 *RevealLtk*(*A*)表示敌手获得了 TCM *A* 的 *PEK* 私钥.该规则允许敌手获得 SM2 密钥协商协议两个参与方的长期密钥,也就是源 TCM 和目标 TCM 的 *PEK* 私钥.

4.3.2 安全属性验证

除了验证协议 2 满足第 4.2 节中的安全属性外,还要验证其满足前向安全性.

定理 3(perfect forward secrecy). $P2 \cup MD_{\Sigma_T} \models^{\forall}_{E_T} \alpha_{OT} \wedge \alpha_{Eq} \Rightarrow \varphi_5$,其中,

$$\begin{aligned}\varphi_5 := \forall A\ B\ mk\ pek\ npk\ y\ tp\ i\ j\ k\ t_1\ t_2.\quad & TCMCreateMigratedBlob(A,B,mk,pk(pek))@i \wedge \\ & TCMConvertMigratedBlob(B,tp,mk,npk,y)@j \wedge \\ & TCMReleaseExchangeSession(y)@k \wedge A \neq B \wedge i \lessdot j \wedge j \lessdot k \wedge \\ & RevealLtk(A)@t_1 \wedge RevealLtk(B)@t_2 \wedge k \lessdot t_1 \wedge k \lessdot t_2 \Rightarrow \\ & \neg(\exists t.\ \ K(mk)@t).\end{aligned}$$

定理 3 表示在假设α_{OT}和α_{Eq}成立的情况下,P2 的所有执行都满足φ_5,即:在以下 3 个条件成立的情况下,协议 1 的所有执行都满足安全属性φ_5:(1) TTP 仅初始化 1 次;(2) 只有目标 TCM 的 *PEK* 证书验证成功时,源 TCM 所有者才授权该 *PEK* 为迁移保护密钥;(3) 只有源 TCM 的 *PEK* 证书验证成功时,目标 TCM 所有者才会允许迁移数据转换至目标 TCM 的存储保护体系中,协议 2 的所有执行都满足安全属性φ_5.安全属性φ_5表示 TCM *A* 在时间点 *i* 以 TCM *B* 的 *pk*(*pek*)为迁移保护密钥创建密钥 *mk* 的迁移包,TCM *B* 在 *i* 之后的时间点 *j* 以任意类型的密钥 *npk* 为新父密钥 *y* 为临时密钥私钥转换了 *mk* 的迁移数据,并在 *j* 之后的时间点 *k* 释放了临时密钥私钥 *y* 对应的密钥协商会话,且 TCM *A* 和 TCM *B* 为不同的 TCM,那么即使敌手在 *k* 之后的时间点获得了 TCM *A* 和 TCM *B* 的 *PEK* 私钥,也不存在时间点 *t*,在该时间点 *mk* 被敌手获得.因此,定理 3 表明:协议 2 执行成功后,即使敌手获得源 TCM 和目标 TCM 的长期密钥,也不能得到之前迁移的密钥.该定理用于验证协议 2 满足前向安全性.

将规则集合 P2 和定理 1~定理 3 的安全属性编码为脚本作为 Tamarin 的输入,经过 Tamarin 验证得出如图 9 所示的结果.

```
==============================================================================
summary of summaries:

analyzed: Tcm_Key_Migration_Protocol2.spthy

  1_can_migrate_key_to_sms4parentkey (exists-trace): verified (19 steps)
  2_can_migrate_key_to_sm2parentkey (exists-trace): verified (19 steps)
  3_migratedkey_security (all-traces): verified (267 steps)
  4_importedkey_security (all-traces): verified (2309 steps)
  5_migration_PFS (all-traces): verified (10084 steps)

==============================================================================
```

Fig.9 Analysis results of the second protocol by Tamarin

图 9 协议 2 的 Tamarin 验证结果

与图 7 中验证结果的输出形式类似,图中第 1 行文字提示后续输出为 Tamarin 的验证结果,第 2 行文字表示验证的协议文件名为 Tcm_Key_Migration_Protocol2.spthy,之后的 5 行输出是协议安全属性的验证结果.图中的

5 行输出分别对应定理 1~定理 3 中φ_1~φ_5的验证结果.与协议 2 预期的安全属性一致,φ_1~φ_5均验证为真.

5 相关工作

在可信安全芯片密钥迁移方面,研究人员对 TPM 密钥迁移机制的研究较多,但目前还没有 TCM 密钥迁移机制的相关研究成果.

在 TPMv1.2 芯片密钥迁移机制的分析方面,文献[16]用一阶逻辑语言建立 TPM API 的形式化模型对 TPM API 进行了全面的逻辑推理分析,分析结果给出了针对 TPM 密钥迁移和授权机制的多种攻击,其中,对密钥迁移机制的攻击使 TPM 的所有者可以破解 TPM 内部存储的其他用户拥有的密钥使用授权信息.文献[17]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,简称 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,但是 TPM 规范对该迁移模式的设计思想是:迁移密钥的保护方式由 MA 决定,即 MA 可以制定自己的安全策略决定对迁移密钥的保护强度,因此,强制 MA 使用 TPM 芯片处理迁移数据会违背该原则.文献[18]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.文中给出的针对密钥迁移类接口的攻击与本文图 3 中的两种攻击类似,导致该类攻击的原因同样是由于缺少源 TPM 和目标 TPM 间的身份认证.但是由于 TPM 缺少具有身份标识作用的加解密密钥,因此目前该类攻击在 TPM 密钥迁移过程中难以避免.

2012 年,可信计算组织发布了 TPM2.0 规范[3],虽然目前还没有符合该规范的芯片产品,但是学术界已经对该规范展开了研究.在 TPM2.0 规范中,密钥迁移被包含在密钥复制机制中.文献[19]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[20]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.

6 结束语

本文首次对 TCM 密钥迁移协议进行了研究和分析,指出现有以新父密钥为迁移保护密钥的密钥迁移协议应用到 TCM 时存在的两个问题:密钥不能被迁移到目标 TCM 的对称存储密钥下;参与协议的两个 TCM 间缺乏相互认证,敌手能够利用该安全缺陷获得源 TCM 的被迁移密钥和将其控制的密钥迁移到目标 TCM 中.针对上述问题,本文提出两个新的密钥迁移协议:协议 1 遵循 TCM 接口规范,允许被迁移密钥以目标 TCM 上任意类型的存储密钥为新父密钥,并解决了源 TCM 对目标 TCM 的身份认证问题;协议 2 需要简单修改 TCM 接口,不仅能完全解决上述两个问题,而且具有前向安全性.本文对上述两个协议进行了形式化分析,论证了协议的正确性并验证了协议满足预期的安全属性.

本文为 TCM 密钥迁移机制提供了新的思路,有助于下一代 TCM 芯片的设计.目前,很多研究工作都针对 TPM 芯片展开,而我国具有自主知识产权的 TCM 芯片具有许多不同于 TPM 的安全特性,同样是一个值得深入研究的对象.我们的下一步工作是研究 TCM 授权协议、TCM 对称密钥管理和其他各种安全机制是否安全、是否存在问题,以完善和改进 TCM.

References:

[1] Feng DG, Qin Y, Wang D, Chu XB. Research on trusted computing technology. Journal of Computer Research and Development, 2011,48(8):1332−1349 (in Chinese with English abstract).

[2] Trusted Computing Group. TPM main specification level 2 version 1.2, revision 116. 2011. http://www.trustedcomputinggroup.org

[3] Trusted Computing Group. Trusted platform module library specification, family “2.0”, level 00, revision 01.07. 2014. http://www.trustedcomputinggroup.org

[4] State Cryptography Administration. GM/T 0011-2012 functionality and interface specification of cryptographic support platform for trusted computing. 2007 (in Chinese). http://www.oscca.gov.cn

[5] State Cryptography Administration. GM/T 0003-2012 public key cryptographic algorithm SM2 based on elliptic curves. 2010 (in Chinese). http://www.oscca.gov.cn

[6] State Cryptography Administration. GM/T 0004-2012 SM3 cryptographic Hash algorithm. 2010 (in Chinese). http://www.oscca.gov.cn

[7] State Cryptography Administration. GM/T 0002-2012 SM4 block cipher algorithm. 2010 (in Chinese). http://www.oscca.gov.cn

[8] China Information Security Standardization Technical Committee. GB/T 25056-2010 information security techniques—Specifications of cryptograph and related security technology certificate authentication system. Beijing: China Zhijian Publishing House, 2010 (in Chinese).

[9] Bruschi D, Cavallaro L, Lanzi A, Monga M. Replay attack in TCG specification and solution. In: Proc. of the 21st Annual Computer Security Applications Conf. New York: IEEE, 2005. 127−137. [doi: 10.1109/CSAC.2005.47]

[10] State Cryptography Administration. GM/T 0012-2012 trusted computing—Interface specification of trusted cryptography module. Beijing: China Zhijian Publishing House, 2012 (in Chinese).

[11] Schmidt B, Meier S, Cremers C, Basin D. Automated analysis of Diffie-Hellman protocols and advanced security properties. In: Proc. of the 25th IEEE Computer Security Foundations Symp. New York: IEEE, 2012. 78−94. [doi: 10.1109/CSF.2012.25]

[12] Meier S, Schmidt B, Cremers C, Basin D. The TAMARIN prover for the symbolic analysis of security protocols. In: Proc. of the 25th Int’l Conf. on Computer Aided Verification. New York: Springer-Verlag, 2013. 696−701. [doi: 10.1007/978-3-642-39799-8_48]

[13] Feng DG. Trusted Computing-Theory and Practice. Beijing: Tsinghua University Press, 2013. 36−47 (in Chinese).

[14] LaMacchia B, Lauter K, Mityagin A. Stronger security of authenticated key exchange. In: Proc. of the 1st Int’l Conf. on Provable Security. New York: Springer-Verlag, 2007. 1−16. [doi: 10.1007/978-3-540-75670-5_1]

[15] Xu J, Feng DG. Comments on the SM2 key exchange protocol. In: Proc. of the 10th Int’l Conf. on Cryptography and Network Security. New York: Springer-Verlag, 2011. 160−171. [doi: 10.1007/978-3-642-25513-7_12]

[16] Chen J. Security analysis of trusted platform module and application [Ph.D. Thesis]. Beijing: Institute of Computing Technology, the Chinese Academy of Sciences, 2006 (in Chinese with English abstract).

[17] Xu SW, Zhang HG. Formal security analysis on trusted platform module based on applied π calculus. Journal of Computer Research and Development, 2011,48(8):1421−1429 (in Chinese with English abstract).

[18] Zhang QY, Zhao SJ, Feng DG. Security analysis and research on TPM migratable key. Journal of Chinese Computer Systems, 2012, 33(10):2188−2193 (in Chinese with English abstract).

[19] Shao JX, Feng DG, Qin Y. Type-Based analysis of protected storage in the TPM. In: Proc. of the 15th Int’l Conf. on Information & Communications Security. New York: Springer-Verlag, 2013. 135−150. [doi: 10.1007/978-3-319-02726-5_11]

[20] Zhang QY, Zhao SJ, Qin Y, Feng DG. Formal analysis of TPM2.0 key management APIs. Chinese Science Bulletin, 2014,59: 4210−4224. [doi: 10.1007/s11434-014-0575-0]

附中文参考文献:

[1] 冯登国,秦宇,汪丹,初晓博.可信计算技术研究.计算机研究与发展,2011,48(8):1332−1349.

[4] 国家密码管理局.GM/T 0011-2012 可信计算密码支撑平台功能与接口规范.2007. http://www.oscca.gov.cn

[5] 国家密码管理局.GM/T 0003-2012 SM2 椭圆曲线公钥密码算法.2010. http://www.oscca.gov.cn

[6] 国家密码管理局.GM/T 0004-2012 SM3 密码杂凑算法.2010. http://www.oscca.gov.cn

[7] 国家密码管理局.GM/T 0002-2012 SM4 分组密码算法.北京:中国质检出版社,2012.

[8] 全国信息安全标准化技术委员会.GB/T 25056-2010 信息安全技术——证书认证系统密码及其相关安全技术规范.北京:中国质检出版社,2010.

[10] 国家密码管理局.GM/T 0012-2012 可信计算可信密码模块接口规范.北京:中国质检出版社,2012.

[13] 冯登国.可信计算——理论与实践.北京:清华大学出版社,2013.36–47.
[16] 陈军.可信平台模块安全性分析与应用[博士学位论文].北京:中国科学院计算技术研究所,2006.
[17] 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.计算机研究与发展,2011,48(8):1421–1429.
[18] 张倩颖,赵世军,冯登国.TPM 可迁移密钥安全性分析与研究.小型微型计算机系统,2012,33(10):2188–2193.

张倩颖(1986－),女,河北三河人,博士,主要研究领域为网络与系统安全,可信计算.

冯登国(1965－),男,博士,研究员,博士生导师,CCF 高级会员,主要研究领域为网络与信息安全.

赵世军(1985－),男,博士,主要研究领域为网络与系统安全,可信计算.