基于Merkle树的TPM单一密钥撤销

余发江,申 淦,张焕国
(武汉大学国家网络安全学院空天信息安全与可信计算教育部重点实验室,湖北武汉 430040)

摘 要: 可信平台模块(Trusted Platform Module,TPM)内部存储空间有限,TPM生成的密钥绝大部分并不会存储于较为安全的TPM内部,而是经过父密钥加密之后再存储于外部存储空间,不完全受TPM控制.在单一密钥无效的情况下,TPM1.2和TPM2.0规范中未提供相关命令来撤销该密钥,只提供了撤销所有密钥的命令,这在多数情况下不方便且降低了TPM的可用性.但是如果不撤销该无效的密钥,攻击者可能会将其加载到TPM中使用,会带来安全隐患.因此,本文基于Merkle树提出了一种能进行单一密钥撤销的密钥管理方案.通过构建动态或者静态Merkle树的方式,将TPM生成的密钥链接到树的叶结点进行密钥管理,在需要的时候可撤销单一无效密钥而不会影响其他有效密钥的正常使用.与基于黑白名单撤销TPM密钥的方案相比,在本文方案中,TPM内部仅需额外保存树的根结点,其余结点存储于TPM的外部,该方案的开销与树能管理的密钥数成对数关系,而黑白名单方案的开销则与被撤销密钥或者未被撤销密钥数量成线性关系;与基于变色龙散列函数构建树来撤销TPM密钥的方案相比,本文的方案更加简便,降低了计算的复杂性.本文基于TPM2.0模拟器构建了一个原型系统,经测试达到了预期目标,具备较好的实用性.

关键词: 可信平台模块;密钥撤销;Merkle树;TPM模拟器
基金项目: 国家自然科学基金(No.61772384)
中图分类号: TP309 **文献标识码:** A **文章编号:** 0372-2112(2023)04-0792-09
电子学报URL:http://www.ejournal.org.cn **DOI:**10.12263/DZXB.20211638

Single Key Revocation Based on Merkle Tree for TPM

YU Fa-jiang,SHEN Gan,ZHANG Huan-guo
(*Key Laboratory of Aerospace Information Security and Trusted Computing (Ministry of Education), School of Cyber Science and Engineering, Wuhan University, Wuhan, Hubei* 430040, *China*)

Abstract: As the internal storage space of trusted platform module (TPM) is limited, most of the keys generated by TPM will not be stored inside TPM. Instead, these keys are encrypted by their parent keys and then stored in external storage space, which is not completely controlled by TPM. In case that one single key is invalid, TPM1.2 and TPM2.0 specifications do not provide any commands to revoke this single key except the command to revoke all keys, which is inconvenient in most cases and reduces the availability of TPM. But if the invalid key is not revoked, an attacker may load it into the TPM for use and this will result in some security issues. Thus, this paper proposes a scheme based on Merkle tree to revoke single key. By constructing dynamic or static Merkle tree, the keys generated by TPM are linked to leaves of the tree for key management, which can revoke single invalid key if needed without affecting the normal use of other keys. Compared with the scheme based on blacklist and whitelist to revoke single key for TPM, our scheme shows only the root of the tree is stored inside the TPM and the remaining nodes are stored outside the TPM. The cost of the scheme has a logarithmic relationship with the number of keys managed by the tree, while the cost of the scheme based on blacklist and whitelist is linear to the number of revoked or unrevoked keys. Compared with the scheme that constructs a tree based on chameleon hash function to revoke single key for TPM, our scheme is simpler and reduces the calculation complexity. This paper builds a prototype system based on TPM2.0 simulator. Through testing, the system achieves the expected goal and has good practicability.

Key words: trusted platform module; key revocation; Merkle tree; TPM simulator
Foundation Item(s): National Natural Science Foundation of China (No.61772384)

收稿日期:2021-12-09;修回日期:2022-08-05;责任编辑:李勇锋

1 引言

可信计算技术目标是提高计算机系统的安全性和确保数据的完整性,其基本思想是以可信平台模块TPM(Trusted Platform Module)为信任根,在计算机中构建了一条信任链,沿着信任链从信任根开始到操作系统再到应用软件,一级信任一级[1,2]. TPM主要以硬件芯片形式出现,来保证计算的安全性与可信性[3,4]. 可信计算平台及TPM已经得到了广泛的应用,例如:Microsoft在Windows操作系统上集成了BitLocker[5],它使用TPM帮助保护Windows操作系统和用户数据,同时最新的Windows操作系统Windows11也要求电脑支持TPM2.0版本[6];Oracle为SPARC T4系统和Oracle Solaris 11等服务器装载了TPM来增强数据的安全性[7];Google Cloud上的安全强化虚拟机利用了虚拟可信平台模块(virtual Trusted Platform Module,vTPM)来保护企业工作负载免遭远程攻击,确保工作负载可靠且可验证[8].

然而,TPM目前在密钥管理方面具有缺陷. 为了便于扩展且TPM内部存储空间有限,将其生成的大部分密钥以加密形式存储于TPM外部,当某一外部密钥泄露时,TPM1.2和TPM2.0规范均未提供命令撤销单一密钥;同时,由于这些存储在外部的密钥可能有多个副本且TPM内部并未保存这些外部密钥的信息,即使TPM提供命令撤销某一副本,也无法彻底撤销该密钥. 若不对其进行撤销,即便使用该密钥的某一应用知道密钥已作废并不再使用,但其他应用可能仍保留该密钥的副本并能通过TPM2.0的验证,违规使用密钥,带来信息安全上的隐患. 目前,TPM2.0规范中提供了TPM2_Clear命令清除所有密钥,这在多数情况下极为不便,降低了TPM的可用性. 因此,对于泄露的单一密钥,TPM目前未提供较好的解决办法,需要一种有效撤销单一密钥的方案.

本文主要针对TPM2.0规范中密钥管理方面的不足,提出一种单一密钥撤销方案,并结合TPM2.0模拟器实现了原型系统,对其性能等方面进行了测试分析. 本文主要贡献有:(1)基于Merkle二叉树提出了一种单一密钥撤销方案,未撤销密钥的散列值以及撤销密钥与撤销标志拼接后的值的散列值各自存放于叶结点中,保存树的根结点于TPM内部以方便验证密钥有效性;(2)采用动态Merkle树与静态Merkle树两种不同的构造方式实现了密钥撤销,两种构造方式具有各自的特点,动态Merkle树能随着密钥逐渐增多而拓展,静态Merkle树则在构建之初分配好了预定的结点空间;(3)结合TPM2.0模拟器实现了该单一密钥撤销方案,进行了原型系统的性能测试,相比于原生TPM模拟器,对TPM内部命令带来了较少的额外时间开销.

2 背景及相关工作

2.1 TPM密钥管理及缺陷

由于TPM内部存储空间有限,对密钥的管理主要是依靠密钥之间的层次结构. 在一个层次结构中,TPM使用种子并执行TPM2_CreatePrimary创建主密钥,主密钥执行TPM2_Create创建并加密子密钥,子密钥又可以当作父密钥加密下一层密钥. TPM2.0有三个持久性hierarchy:平台、存储和背书[9,10],它们每一个都有一个种子,用于创建各自的主密钥. 在TPM2.0中,TPM内部存储种子,而基于种子生成的主密钥以及其余由父密钥加密的子密钥,大部分存放于外部存储空间.

然而,在某一外部密钥已经被泄露的情况下,TPM2.0并未提供任何命令进行单一密钥的撤销,只能使用TPM2_Clear清除所有密钥,影响其它未泄露密钥的使用.

2.2 相关工作

2.2.1 TPM相关研究

近年来,与TPM相关的研究主要包含了密钥隐私泄露[11]、授权认证机制分析[12]、移动设备及嵌入式设备TPM方案分析[13,14]、远程认证[15]等方面.

2.2.2 TPM单一密钥撤销

Katzenbeisser等人为TPM1.2提出了两种方案[16]:黑名单和白名单. 在黑名单方案中,黑名单是一个存放于外部的散列链表,用于记录被撤销了的密钥. 当密钥失效需要被撤销时,就将该密钥记录到黑名单中,当加载某一密钥时,将黑名单中的密钥按顺序发送给TPM,若是该密钥在这些被撤销的名单中,TPM将放弃加载并认为该密钥无效. 因此,黑名单方案的开销与被撤销密钥数量成线性关系. 在白名单方案中,在所有外部存放的密钥块中保存TPM内部计数器的散列值,每当有密钥被撤销,TPM将内部计数器散列值递增并且同步更新所有未被撤销密钥保存的散列值. 加载某一密钥时,仅当该密钥保存的散列值与TPM内部计数器的散列值一致才表明该密钥有效. 由此可见,白名单方案的开销与未被撤销密钥数量相关成线性关系.

Yu等人[17]为TPM2.0提出了二叉树与变色龙散列函数相结合的密钥管理方案. 这棵树从叶结点开始自底向上生长,最左侧叶结点保存密钥的变色龙散列值,其余左叶结点保存密钥的普通散列值,右叶结点保存密钥的变色龙散列值. 整棵树的结构类似Merkle二叉树,不同的是,对于非叶结点,最左侧结点保存其两个子结点值拼接后的变色龙散列值,其余左结点保存其两个子结点值拼接后的普通散列值,右结点保存其两个子结点值拼接后的变色龙散列值. 将这棵树的所有最左侧结点(包括树根结点)保存在TPM内部用于后续的密钥验证,其余结点保存于TPM外部便于减少TPM

内部存储空间的开销. 当有密钥添加时,若树中存在对应叶结点,则更新该叶结点的值,若树中不存在对应叶结点,则新生成一个叶结点的值. 计算完叶结点后,更新对应路径上的结点值. 该方案通过更新无效密钥对应叶结点值及相应路径上结点值来撤销单一密钥,当已撤销的密钥再次加载时,沿着路径计算对应最左侧结点的值并与保存在TPM内部的值对比,值不匹配表明该密钥已被撤销. 其优点就是利用了变色龙散列函数的陷门碰撞特性,在因添加密钥或者撤销密钥带来树中结构的改变时,可以减少计算量;而且由于树形结构特性,只需在TPM内部保存最左侧结点,即可实现指数倍数的密钥管理. 但是该方案中变色龙散列函数比普通散列函数的计算时间长.

2.3 Merkle树

Merkle树是Merkle于1989年提出的一种树形结构[18],常用于对数据进行完整性验证. 目前,Merkle树的应用包含了在区块链中确保交易数据的完整性与不可篡改性[19~23]、智能合约[24]以及在云服务器中确保数据完整性等[25].

3 方案概述及威胁分析

3.1 方案概述

本文基于Merkle树提出了一种TPM单一密钥撤销方案,其系统模型如图1所示,该系统主要由TPM内部的密钥链接处、TPM软件栈(TPM Software Stack,TSS)中的结点传输处、应用层的结点处理处以及存储于外部的结点相关文件组成. 本文采用了两种不同的Merkle树构造方式实现该方案,仅存储树的根结点于TPM内部,便于节省其存储开销;其余结点存储于TPM外部的结点相关文件,包括根结点副本,但仅以TPM内部根结点为准.

该系统为密钥提供了三种操作:添加、验证和撤销. TPM生成新的密钥时,进行添加操作将其链接到Merkle树的叶结点上,计算该叶结点到根结点路径上所有结点的散列值,并存储根结点值. TPM加载密钥时,沿着该密钥所在路径计算根结点值,并与存储在TPM内部的根结点值对比,返回密钥是否有效的结果. TPM进行密钥撤销前,先进行加载密钥操作,成功加载密钥后再进行撤销,更新该密钥所在路径上所有结点的散列值.

该系统提供的三种操作是由其几个组成部分共同完成的,结点处理处负责:(1)发出三种密钥相关操作的请求并进行预处理;(2)根据请求从结点相关文件读取树的结点以及将结点写入该文件. 密钥链接处负责:(1)将TPM新生成的密钥链接到树的叶结点进行密钥的散列值计算并将树中待更新的相关结点值传出

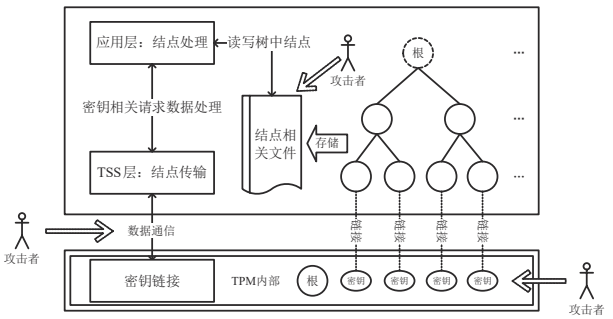

图1 系统模型

TPM;(2)存储Merkle树的根结点于TPM内部用于验证密钥有效性. 结点传输处则负责TPM内部密钥链接处和外部结点处理处之间的数据传输及解析.

3.2 安全目标及假设与威胁分析

本文仅考虑密钥撤销相关的安全问题,提出的方案的安全目标为:(1)在密钥被泄露的情况下,能撤销该无效密钥,无法用其它密钥进行仿冒撤销;(2)当无效密钥被撤销之后,TPM无法加载该密钥,避免带来安全隐患.

该方案有以下几点安全假设和威胁分析.

假设1 存储于TPM内部的根结点始终是安全的,是不受到外部威胁的,攻击者无法获取TPM内部相关数据.

假设2 所选取散列函数在计算上是安全的. 令Hash()表示所用散列函数. 其一,给定散列值K,攻击者无法计算出满足Hash(m)=K的m. 其二,给定消息m,攻击者无法计算出满足Hash(m')=Hash(m)的m'.

假设3 进行密钥相关操作时,传入到TPM内部的路径顺序是正确的.

威胁分析如下.

威胁1 TPM生成密钥并将其存储于外部之后,攻击者可能篡改外部的密钥数据.

威胁2 攻击者可能窃取并篡改外部的结点相关文件中存储的树的结点信息.

4 TPM单一密钥撤销方案

4.1 具备单一密钥撤销功能的Merkle树构造方式

4.1.1 动态Merkle树

本文方案所采用的第一种Merkle树构造方式是基于二叉树的动态Merkle树,叶结点所保存的密钥数量不确定,树的规模随着密钥的数量增加而增大. 这种构造方式生成的树不一定是满二叉树,即其所有叶结点并不一定是在最下层. 只要是树中的叶结点,均保存对应密钥的散列值.

树的结点信息中包含了结点索引和结点值,除了传统Merkle树所具有的性质之外,结点索引值还具有

二分搜索树的性质,规则如下:(1)结点索引为从1开始的连续正整数;(2)所有叶结点的索引为连续正奇数,包括这棵树不是满二叉树的情况.

图2和图3展示了树的高度为4时,满二叉树和非满二叉树的情况下,动态Merkle树中包含的结点的索引分布,本文用v_i表示索引值为i的结点值及该结点本身.

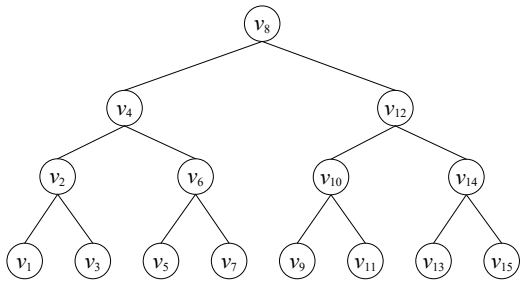

图2　动态Merkle二叉树满树

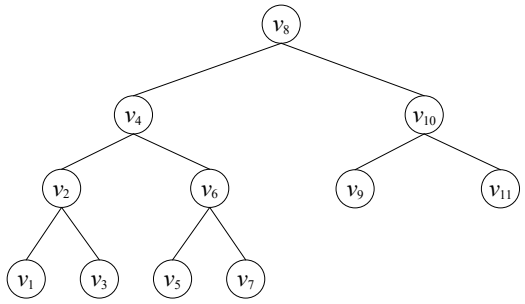

图3　动态Merkle二叉树非满树

4.1.2　静态Merkle树

与动态Merkle树不同,本文所采用的第二种Merkle树构造方式是基于满二叉树的静态Merkle树,这棵树在初始化时便自底向上构建为一棵满二叉树.树所能管理的最大密钥数量是确定的,其规模不会随着密钥的数量增加而增大,所有的叶结点均在同一层.

这种构造方式不具有动态Merkle树构造方式中结点索引间的关系,将采取新的规则,具体如下:

(1)结点索引采用二维索引(Height, Index),包含结点高度以及在每一层的序号,结点高度与序号均从0开始计算;

(2)密钥可以链接到初始化后任意未使用的叶结点上.

图4展示了高度为4时,已预先构建好的静态Merkle树中的结点索引分布.

4.2　获取认证结点

使用Merkle树对密钥进行添加、撤销及验证操作时,都涉及到与该密钥所对应叶结点到树的根结点之间相关结点值的插入、更新或者读取,采用获取认证结点算法来得到这些结点的集合.

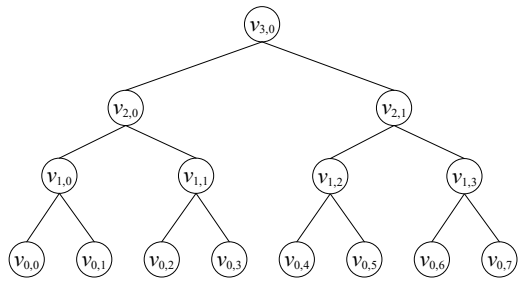

图4　静态Merkle二叉树

算法输入为密钥对应叶结点索引和树的根结点索引.算法输出为从树中读取的结点集合inSet,以及树中需要进行插入值、更新值的结点集合outSet,其流程见算法1.

算法1　GenAuthSet

输入:leafIndex, rootIndex

输出:inSet, outSet

1.根据不同树的结构获取leaf到root的唯一路径

2.将路径上的结点索引加入outSet

3.将路径上的结点的兄弟结点索引加入inSet

该算法因两种树的结构不同而产生的差异如下.

(1)动态Merkle树

由于动态Merkle密钥管理树具有二叉搜索树的性质,GenAuthSet从树的根结点开始.每一轮循环将当前的根结点索引加入outSet.然后比较当前根结点索引和密钥对应叶结点索引之间大小关系,若前者小于后者,则将当前根结点左子树根结点索引加入inSet,再将当前根结点索引赋值为其右子树根结点索引;反之则将当前根结点右子树根结点索引加入inSet,并将当前根结点索引赋值为其左子树根结点索引.重复此操作,直至当前根结点索引与密钥对应叶结点索引相等,停止循环,将叶结点索引加入outSet.

(2)静态Merkle树

静态Merkle树的结点索引包含了结点的高度和序号信息,GenAuthSet从叶结点开始,每一轮循环将当前的叶结点索引加入outSet,根据兄弟结点序号间的奇偶关系计算出其兄弟结点索引并加入inSet,再根据叶结点的高度和当前层序号计算出父结点索引,重复以上操作,直至将根结点索引加入outSet.

4.3　更新结点集合

在进行密钥添加、撤销及验证操作时,需要根据获取认证结点算法读取的结点集合执行更新结点集合算法更新相应的结点集合outSet,最后再根据各个操作实际需求决定是否将outSet中的结点更新到外部结点相关文件.

算法的输入为inSet和outSet,inSet为需要读取的

结点集合，包含结点索引和结点值，在算法执行过程中不会对其进行更改；outSet 为需要更新值的结点集合，在作为输入时，其中部分结点可能仅包含结点索引，其流程见算法 2.

算法 2 UpdateAuthSet

输入：inSet,outSet

输出：outSet

1. 根据路径上的顺序从叶结点开始读取 outSet 中一个结结点，并从 inSet 中读取其兄弟结点
2. 根据两结点索引将其分为左结点 left 与右节点 right
3. 计算 outSet 中下一个结点值为 Hash(left||right)
4. 重复以上操作直至按序计算完所有输出结点值

4.4 添加密钥

当 TPM 需要将密钥添加到 Merkle 树里面进行密钥管理时，执行添加操作，算法的输入为叶结点索引、根结点索引以及密钥，当采用动态 Merkle 树时，需要输入已添加密钥数量 c，其流程见算法 3.

算法 3 Append

输入：leafIndex, rootIndex, key, c(若是动态 Merkle 树)

输出：outSet

1. 静态 Merkle 树执行 GenAuthSet(leafIndex,rootIndex),动态 Merkle 树执行 GenAuthSet($2\cdot c-1$,rootIndex)且 $c\leftarrow c+1$,得到 inSet,outSet
2. 计算输出结点集合中叶结点值为 Hash(key)
3. 执行 UpdateAuthSet(inset,outSet)

该算法因两种树的结构不同，对动态 Merkle 树需进行差异化处理. 根据已添加密钥数量 c 的不同，可将操作分为两类.

(1)若 c=0，则为当前添加的密钥生成一个根结点并将其索引设为 1，值设为密钥的散列值，之后 c 的值加 1，并存储根结点于 TPM 内部.

(2)若 c>0，且当前树为满二叉树时，先要对其进行拓展，即将根结点索引翻倍来实现结点容量扩容. 然后已添加密钥数量 c 增加 1，由该 Merkle 密钥管理树具有二叉搜索树性质，结点处理处计算出待添加密钥对应叶结点索引为 $2\cdot c-1$. 执行 GenAuthSet 算法得到需要读入、更改的结点集合，根据 inSet 中结点索引读取外部结点相关文件获取结点值，在密钥链接处计算密钥对应叶结点的散列值并结合 inSet 各个结点的值对 outSet 中结点进行值的插入或者更新. 最后将根结点存储于 TPM 内部，outSet 中其余结点存于外部结点相关文件，完成添加操作.

图 5 展示了添加前 4 个密钥时动态 Merkle 树的构建过程. 已添加密钥数量 c 为 0 时，如图 5(a)所示，生成仅有一个根结点 v_1 的树，密钥 1 链接到 v_1 上并计算 v_1 的值. 添加密钥 2 时如图 5(b)所示，由于当前树已满，先拓展树的根结点索引为 2，再生成新叶结点 v_3 来链接密钥 2，并计算 v_2 的值. 添加密钥 3 时如图 5(c)所示，先拓展根结点索引为 4，生成新的叶结点 v_5 来链接密钥 3，并计算 v_4 的值. 添加密钥 4 时如图 5(d)所示，根据叶结点索引为连续奇数规律，生成新叶结点 v_7 来链接密钥 4，并更新 v_4 的值.

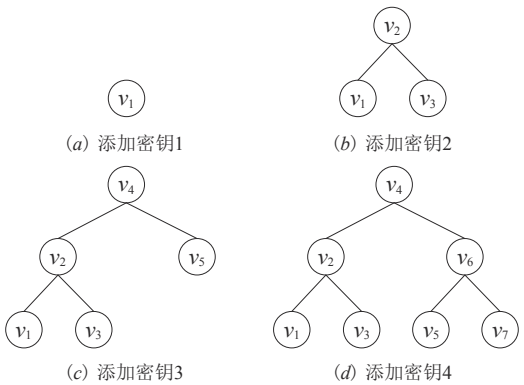

图 5 添加 4 个密钥构建动态 Merkle 树的过程

对于静态 Merkle 树，无需在添加密钥过程中拓展树的结构，密钥可链接到任意叶结点. 根据输入的根结点索引及密钥链接的叶结点索引，执行 GenAuthSet 得到路径信息并读取结点值后，在密钥链接处进行 UpdateAuthSet，更新树的结构并存储相关结点即可.

4.5 撤销密钥

撤销单一密钥时，其流程见算法 4. 结点处理处根据当前根结点索引以及要撤销密钥对应叶结点索引，执行获取认证结点算法得到需要读取及更改的结点集合. 然后密钥链接处在密钥块中加上撤销标志 REVOKE_FLAG，重新计算叶结点散列值，结合读取的 inSet 对 outSet 中各结点值进行更新，最后存储相关结点值，撤销结束.

算法 4 Revoke

输入：leafIndex, rootIndex, key

输出：outSet

1. 执行 GenAuthSet(leafIndex,rootIndex)得到 inSet,outSet
2. 更新 outSet 中叶结点值为 Hash(key||REVOKE_FLAG)
3. 执行 UpdateAuthSet(inSet,outSet)

4.6 验证密钥

当验证密钥有效性时，其流程见算法 5. 结点处理处根据根结点索引以及被验证密钥对应叶结点索引，执行获取认证结点算法得到要读取的结点集合. 然后密钥链接处计算密钥散列值，结合读取的 inSet 各结点值在内存中对 outSet 各结点值进行更新，最后计算出一个临时使用的根结点值与保存在 TPM 内部的根结点值对比，若两值相同则返回密钥有效的结果. 若两值不

同,则将密钥加上撤销标志REVOKE_FLAG之后计算密钥散列值并重新计算临时使用的根结点值,将该值与保存在TPM内部的根结点值对比,若两值相同则表明该密钥已经被撤销,返回密钥无效的结果;若两值不同则表明该密钥由于未知原因无效,验证结束.

算法5 Verify

输入:leafIndex, rootIndex, key

输出:密钥有效性

```
1.执行GenAuthSet(leafIndex,rootIndex)得到inSet,outSet
2.计算outSet中叶结点值为Hash(key)
3.执行UpdateAuthSet(inSet,outSet)
4.if outSet中根结点值与TPM内部值相等 then
5.  密钥有效
6.else
7.  计算outSet中叶结点值为Hash(key||REVOKE_FLAG)
8.  UpdateAuthSet(inSet,outSet)
9.  if outSet中根结点值与TPM内部值相等 then
10.   密钥已撤销
11. else
12.     未知无效密钥
13. end if
14.end if
```

5　安全性分析

本章根据3.2的安全假设与威胁分析,列出攻击者可能进行的攻击行为,基于被撤销的无效密钥是否存在安全隐患的角度,分析这些行为是否会对该方案造成威胁.

(1)行为1:若密钥key已被密钥持有者向TPM发出撤销申请,攻击者持有密钥副本并且想保留该密钥key的可用性. 在TPM进行撤销操作之前,假设结点相关文件中该密钥对应叶结点值为v_1,根结点值为v_r,即:v_1=Hash(key).

攻击者若是在外部修改密钥key的数据,企图用篡改过的密钥数据key′传入到TPM内部代替key进行仿冒撤销,使计算出来的根结点值v_r′与未撤销时保持一致,当加载原密钥key时便可通过验证. 假设撤销完成后该密钥对应叶结点值为v_1′,则:v_1′=Hash(key′||REVOKE_FLAG).

要达到攻击者的目的,则需v_1=v_1′,在所选散列函数具有单向性与抗碰撞性的前提下,需满足key=key′||REVOKE_FLAG,即修改后的密钥与撤销标志位的拼接值要与原密钥相同. 在攻击者无法计算撤销标志位的情况下,使得key=key′||REVOKE_FLAG是困难的,因为攻击者无法得知撤销标志位长度;即使在标志位长度正确情况下,删除原密钥key末尾一定长度的数据得到key′,拼接上REVOKE_FLAG,但由于密钥数据采取二进制方式,可通过在撤销标志位中加上二进制密钥数据中不可能出现的字符,或者使撤销标志位长度大于密钥长度,使得拼接后的数据无法与原密钥key一致. 且存放于外部的TPM密钥是加密的,当攻击者修改了该数据后在执行TPM2_Load过程中解密时是无法通过TPM自带的HMAC认证的. 因此,攻击者企图用篡改后的密钥来代替将被撤销的密钥执行撤销过程是不可行的,原无效密钥仍会被撤销,达到了安全目标.

(2)行为2:若密钥key已被TPM撤销,攻击者持有密钥副本并且想继续加载密钥key. 在TPM进行了撤销操作之后,假设结点相关文件中该密钥对应叶结点值为v_1,即:v_1=Hash(key||REVOKE_FLAG),根据此v_1的值沿着路径更新v_r的值.

攻击者若是在外部修改密钥key的数据,企图用篡改过的密钥数据key′加载到TPM内部代替key进行验证,使计算出来的根结点值v_r′与撤销后保持一致,设加载时密钥链接叶结点值为v_1′,则:v_1′=Hash(key′).

通过验证要满足v_r=v_r′,则需v_1=v_1′. 根据安全假设,TPM内部是安全的,且所选散列函数具有单向性与抗碰撞性,因此需满足key||REVOKE_FLAG=key′,由于密钥拼接撤销标志位是在TPM内部进行,攻击者在不知道撤销标志位长度和内容的情况下强行修改密钥数据使key′=key||REVOKE_FLAG是困难的. 且同行为1中阐述一样,修改密钥数据后无法通过TPM的HMAC认证. 因此,在本文安全假设下,攻击者无法加载被撤销的无效密钥,达到了安全目标.

综上所述,基于安全假设,本文的方案能安全有效地撤销TPM单一密钥,防止TPM使用撤销后的无效密钥,确保系统的安全性.

6　原型系统

本文结合该单一密钥撤销方案,基于开源项目TPM模拟器IBM's Software TPM2.0[26]、Intel's TPM2.0 Software Stack以及Intel's TPM2.0 Tools[27]实现了原型系统. 结合TPM2.0规范,新增树的结点所对应的TPM结构体以及相应的解析函数,并对相关命令与函数进行调整,在TPM内部实现了密钥链接、TSS层面实现了结点传输、提供给应用调用的Tools层面实现了结点处理.

该系统采用的散列函数为SHA256算法,其能产生32个字节的定长摘要,便于统一处理;并选取可扩展标记语言(eXtensible Markup Language,XML)文件在TPM外部存储结点相关信息,XML文件易于读写数据的特性比较适用于存储树的结点信息.

7 性能分析与测试

原型系统运行在具有2 896 MB内存、两个处理器(其中每个处理器具有两个内核),Ubuntu18.04.5 desktop-amd64操作系统的VMware16.1.2的虚拟机上. 物理主机配置为:3.2 GHz的CPU频率,处理器为AMD Ryzen 7 5800H, 16 GB内存,安装了Windows10专业版操作系统.

在两种不同构造方式的树中,实验所测试的密钥数量最大设定为$2^{13}=8\,192$,即在Merkle树中,最大管理的密钥数量为8 192,叶结点最大索引为$2^{14}-1$时,对应的动态Merkle树拓展为一棵高度为14的满二叉树. 基于该设定,在不同叶结点索引区间,实验测试了两种方案下的创建密钥、撤销密钥以及加载密钥的时间开销,并与未改动的原开源项目相比较.

对于动态Merkle树,实验测试了树的高度(从叶结点到根结点最大层数)分别为2、4、6、8、10、12、14时且是满二叉树状态下(各种情况下动态Merkle树能管理的最大密钥数分别为2^1、2^3、2^5、2^7、2^9、2^{11}、2^{13}),对树中最大索引叶结点进行密钥相关操作所带来的影响.

对于静态Merkle树,实验中设定树的高度为14,测试了与动态Merkle树中被测试叶结点相同位置的结点相关的密钥操作,例如:动态Merkle树索引为2^2-1、2^4-1、2^6-1、…、2^k-1的叶结点分别对应静态Merkle树中索引为$(0,1)$、$(0,7)$、$(0,31)$、…、$(0,2^{k-1}-1)$的叶结点,因此实验结果图片中叶节点索引统一为动态Merkle树索引.

7.1 时间开销

在所有的时间开销测试中,未考虑从外部XML文件中读写树的结点信息带来的时间开销,仅将无法避开的IO操作纳入时间开销测试范围. 测试采用搭载了Linux系统的虚拟机的CPU时钟计时单元(clock tick)为单位,1秒为10^6个CPU时钟计时单元. 图6、图7及图8展示了改动后的Tools层各函数和TPM内部命令与未改动时的时间开销对比.

如图6所示,对于Tools层的tpm2_create和tpm2_load函数,未改动的原函数采取多次运行的平均值作为7次的值. 采用动态Merkle树时,改动后函数的时间开销随着树的高度增加而变大. 在树的高度为8及以下时,时间开销与原函数相比处于比较好的范围;树的高度为10或12时,由于叶结点数按幂增长幅度大,时间开销略高,但仍处于可接受范围;当树的高度为14且是满二叉树时,测试叶结点数从2 048陡增到8 192,故带来的时间开销增长幅度较大. 采用静态Merkle树时,由于树的高度和树中的结点数固定,进行密钥操作时,无论该密钥链接到任何索引的叶结点,其所影响的路径上的结点数是一致的,因此带来的时间开销保持在一个水平位置附近. 此外,对于两种树,新增的撤销操作在TPM外部执行的函数tpm2_revoke均与改动后的另外两个函数时间开销接近,达到了预计目标,可以接受.

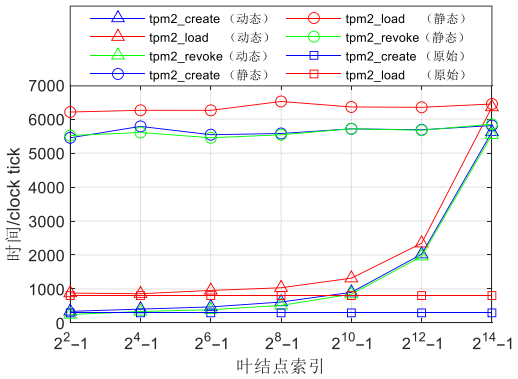

图6 Tools层函数时间开销

如图7所示,TPM内部的TPM2_Create命令本身在创建密钥时耗时较多,且本系统的添加密钥操作在TPM内部需处理两种树的结点数较少,树的每一层仅需读写一结点,因此该原型系统对TPM2_Create命令影响较小,改动后时间开销在原命令的时间开销平均值附近波动.

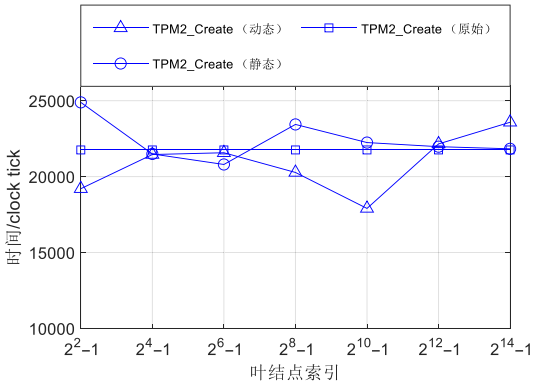

图7 TPM2_Create时间开销

如图8所示,TPM内部的TPM2_Load命令时间开销较小,原命令时间开销平均仅为136个CPU时钟计时单元,因此即便改动后的TPM2_Load命令要额外处理的结点数量级很小,也能明显看出与原命令的差距. 随着动态Merkle树的高度增加,改动后的TPM2_Load命令时间开销增长较为平稳,在可接受范围内,而静态Merkle树的时间开销则在略高于原命令开销的一个范围内波动. 此外,TPM2_Revoke的时间开销表现良好,远小于同为TPM内部命令的TPM2_Create. 因此,根据整个原型系统的时间开销表现,新增的撤销操作对TPM原有的命令影响不大,是可以接受的.

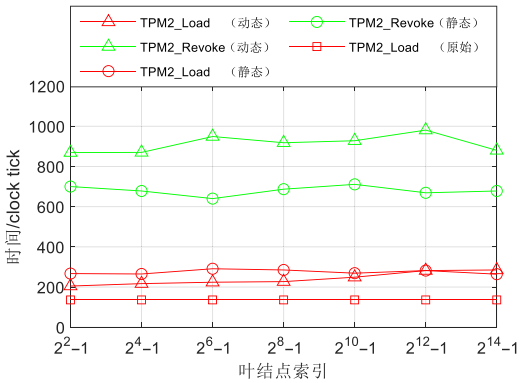

图8　TPM2_Load与TPM2_Revoke时间开销

7.2　存储开销

对于动态Merkle树,TPM生成的密钥越多,树的高度越高,且密钥仅链接到新生成的叶结点上. 若当前动态Merkle树的高度为H,TPM内部仅存储树的根结点,满二叉树时其可链接的密钥数量达到最大值2^{H-1},存储于外部结点相关文件的结点数为2^H-1;最少已链接的密钥数为$2^{H-2}+1$,存储于外部结点相关文件的结点数为$2^{H-1}+1$. 因此对于高度为H的动态Merkle树,在已链接密钥数量最大和最小情况下,"密钥数:外部结点相关文件存储结点数:TPM存储结点数"分别为"2^{H-1}:2^H-1:1"和"$2^{H-2}+1$:$2^{H-1}+1$:1". 可以将该比例分别简化为"2^{H-1}:2^H:1"和"2^{H-2}:2^{H-1}:1",因此,不管已链接密钥是否达到了树中最大数,树中总结点数约为已链接密钥数的两倍,但是TPM内部只需额外存储一个根结点.

对于静态Merkle树,其为高度固定的满二叉树,新密钥可链接到任意未链接密钥的叶结点上. 在高度为H的情况下,"最大密钥数:外部结点相关文件存储结点数:TPM存储结点数"为"2^{H-1}:2^H-1:1";然而当密钥数很小,例如为1时,"密钥数:外部结点相关文件存储结点数:TPM存储结点数"则变为"1:2^H-1:1". 因此,在树的高度已固定的情况下且密钥数量远小于树可链接的最大密钥数时,采取该结构带来的外部多余的存储开销较大.

两种结构中的树的结点值均为256比特的散列值,均采取长度为64字节的字符串存储,因此无论TPM生成的密钥有多少,相比于改动之前,对于TPM内部额外增加的存储开销主要为64字节. 而外部的存储空间不像TPM内部受限制,可以根据实际需求分配存储空间. 因此,该原型系统能以牺牲较少的TPM内部空间来换取对大量密钥的管理.

8　总结与展望

本文提出了一种基于Merkle树的TPM单一密钥撤销方案,采用了两种不同的Merkle树构造方式来实现了该方案,并基于TPM2.0模拟器实现了原型系统,达到了能撤销TPM单一密钥的功能,防止无效密钥可能会对TPM带来的隐患. 该方案中TPM内部需要处理的树的结点数与密钥数之间呈现对数关系,对TPM内部带来的额外存储开销仅为存储树的根结点所需的空间. 后续可考虑在TPM外部的应用与密钥间建立密钥映射表,该映射表保存某应用所持有的密钥序号,以及密钥在树中链接的结点位置,并存储映射表,结合本文方案完善TPM单一密钥撤销机制.

参考文献

[1] 沈昌祥, 张焕国, 王怀民, 等. 可信计算的研究与发展[J]. 中国科学: 信息科学, 2010, 40(2): 139-166.
SHEN Chang-xiang, ZHANG Huan-guo, WANG Huai-min, et al. Research and development of trusted computing [J]. Chinese Science: Information Science, 2010, 40(2): 139-166. (in Chinese)

[2] 张焕国, 赵波. 可信计算[M]. 武汉: 武汉大学出版社, 2011.
ZHANG Huanguo, ZHAO Bo. Trusted Computing[M]. Wuhan: Wuhan University Press, 2011. (in Chinese)

[3] YU Fajiang, ZHANG Huanguo, ZHAO Bo, et al. A formal analysis of trusted platform module 2.0 hash-based message authentication code authorization under digital rights management scenario[J]. Security and Communication Networks, 2015, 9(15): 1-14.

[4] 宋敏, 谭良. 一种TPM2.0密钥迁移协议及安全分析[J]. 电子学报, 2019, 47(7): 1449-1464.
SONG Min, TAN Liang. A TPM2.0 key migration protocol and security analysis[J]. Acta Electronica Sinica, 2019, 47 (7): 1449-1464. (in Chinese)

[5] Corporation Microsoft. Bitlocker overview[EB/OL]. [2021-12-09]. https://docs.microsoft.com/zh-cn/windows/security/information-protection/bitlocker/bitlocker-overview.

[6] Corporation Microsoft. Get-Windows-11[EB/OL]. [2021-12-09]. https://www.microsoft.com/zh-cn/windows/get-windows-11.

[7] Oracle. Oracle solaris and oracle SPARC T4 servers—Engineered together for enterprise cloud deployments[EB/OL]. [2021-12-09]. https://www. oracle. com/us/products/servers-storage/solaris/solaris-and-sparc-t4-497273.pdf.

[8] Cloud Google. Shielded-VM[EB/OL]. [2021-12-09]. https://cloud.google.com/shielded-vm.

[9] Trusted Computing Group. TPM 2.0 library specification part 1 Architecture[EB/OL]. [2021-12-09]. https://trustedcomputin-ggroup.org/wp-content/uploads/TCG_TPM2_r1p59_Part1_Architecture_pub.pdf.

[10] Trusted Computing Group. TPM 2.0 library specification part 3 Commands[EB/OL]. [2021-12-09]. https://trustedcomputinggroup.org/wp-content/uploads/TCG_TPM2_r1p59_Part3_Commands_pub.pdf.

[11] 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, Heinayati Milan, et al. Security enhancement of key duplication in TPM2.0[J]. Journal of Wuhan University (Natural Science Edition), 2014, 60(6): 471-477. (in Chinese)

[12] SHAO J, QIN Y, FENG D. Formal analysis of HMAC authorization in the TPM2.0 specification[J]. IET Information Security, 2018, 12(2): 133-140.

[13] CHAKRABORTY D, HANZLIK L, BUGIEL S. Simtpm: User-centric TPM for mobile devices[C]//28th USENIX Security Symposium (USENIX Security 19). Santa Clara: USENIX Association, 2019: 533-550.

[14] LU D, HAN R, WANG Y, et al. A secured TPM integration scheme towards smart embedded system based collaboration network[J]. Computers Security, 2020, 97: 101922.

[15] DAVE A, WISEMAN M, SAFFORD D. SEDAT: Security enhanced device attestation with TPM 2.0 [EB/OL]. (2021-01-16) [2021-12-09]. https://doi. org/10.48550/arXiv. 2101.06362.

[16] KATZENBEISSER S, KURSAWE K, STUMPF F. Revocation of tpm keys[C]//Trusted Computing. Berlin: Springer Berlin Heidelberg, 2009: 120-132.

[17] 余发江, 陈宇驰, 张焕国. 具备撤销单一密钥功能的TPM动态密钥管理机制[J]. 清华大学学报(自然科学版), 2020, 60(6): 464-473.
YU Fajiang, CHEN Yuchi, ZHANG Huanguo. Dynamic key management mechanism with individual key revocation ability for TPM[J]. Journal of Tsinghua University (Science and Technology), 2020, 60(6): 464-473. (in Chinese)

[18] MERKLE R. A certified digital signature[C]//Crypto 1989. New York: ACM, 218-238.

[19] 邵奇峰, 金澈清, 张召,等. 区块链技术:架构及进展[J]. 计算机学报, 2018, 41(5): 969-988.
SHAO Qi-Feng, JIN Che-Qing, ZHANG Zhao, et al. Blockchain: Architecture and research progress[J]. Chinese Journal of Computers, 2018, 41(5): 969-988. (in Chinese)

[20] 陈露, 相峰, 孙知信. 基于属性密码体制的区块链安全技术研究进展[J]. 电子学报, 2021, 49(1): 192-200.
CHEN Lu, XIANG Feng, SUN Zhi-xin. A survey of blockchain security technologies based on attribute-based cryptography[J]. Acta Electronica Sinica, 2021, 49(1): 192-200. (in Chinese)

[21] 马晓婷, 马文平, 刘小雪. 基于区块链技术的跨域认证方案[J]. 电子学报, 2018, 46(11): 2571-2579.
MA Xiao-ting, MA Wen-ping, LIU Xiao-xue. A cross domain authentication scheme based on blockchain technology[J]. Acta Electronica Sinica, 2018, 46(11): 2571-2579. (in Chinese)

[22] 于戈, 聂铁铮, 李晓华, 等. 区块链系统中的分布式数据管理技术——挑战与展望[J]. 计算机学报, 2021, 44(1): 28-54.
YU Ge, NIE Tie-Zheng, LI Xiao-Hua, et al. The challenge and prospect of distributed data management techniques in blockchain systems[J]. Chinese Journal of Computers, 2021, 44(1): 28-54. (in Chinese)

[23] 秦超霞, 郭兵, 沈艳, 等. 区块链的安全风险评估模型[J]. 电子学报, 2021, 49(1): 117-124.
QIN Chao-xia, GUO Bing, SHEN Yan, et al. Security risk assessment model of blockchain[J]. Acta Electronica Sinica, 2021, 49(1): 117-124. (in Chinese)

[24] BRUSCHI F, RANA V, PAGANI A, et al. Tunneling trust into the blockchain: A merkle based proof system for structured documents[J]. IEEE Access, 2021, 9: 103758-103771.

[25] 陈兰香, 邱林冰. 基于Merkle哈希树的可验证密文检索方案[J]. 信息网络安全, 2017, (4): 1-8.
CHEN Lanxiang, QIU Linbing. A verifiable ciphertext retrieval scheme based on Merkle Hash tree[J]. Netinfo Security, 2017, (4): 1-8. (in Chinese)

[26] IBM. IBM's Software TPM2.0[CP/OL]. [2021-12-09]. https://sourceforge.net/projects/ibmswtpm2/.

[27] Linux TPM2 & TSS2 Software[CP/OL]. [2021-12-09]. https://github.com/tpm2-software.

作者简介

余发江 男,1980年出生于重庆市,2007年毕业于武汉大学计算机学院信息安全专业,获博士学位. 现为武汉大学国家网络安全学院副教授,硕士生导师. 主要从事系统安全、可信计算等方面的教学和研究工作.
E-mail: fjyu@whu.edu.cn

申 淦 男,1998年6月出生于贵州毕节,现为武汉大学国家网络安全学院硕士. 主要研究方向为可信计算.
E-mail: 2016301200081@whu.edu.cn