计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2023,32(9):53−66 [doi: 10.15888/j.cnki.csa.009228]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

基于 DICE 的证明存储方案①

王　辉[1,2], 冯　伟[2], 秦　宇[2]

[1](中国科学院大学, 北京 100049)
[2](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)
通信作者: 冯　伟, E-mail: fengwei2009@iscas.ac.cn

摘　要: 信息技术的不断发展和智能终端设备的普及导致全球数据存储总量持续增长, 数据面临的威胁挑战也随着其重要性的凸显而日益增加, 但目前部分计算设备和存储设备仍存在缺乏数据保护模块或数据保护能力较弱的问题. 现有数据安全存储技术一般通过加密的方式实现对数据的保护, 但是数据的加解密操作即数据保护过程通常都在应用设备上执行, 导致应用设备遭受各类攻击时会对存储数据的安全造成威胁. 针对以上问题, 本文提出了一种基于 DICE 的物联网设备证明存储方案, 利用基于轻量级信任根 DICE 构建的可信物联网设备为通用计算设备 (统称为主机) 提供安全存储服务, 将数据的加解密操作移至可信物联网设备上执行, 消除因主机遭受内存攻击等风险对存储数据造成的威胁. 本文工作主要包括以下 3 方面: (1) 利用信任根 DICE 构建可信物联网设备, 为提供可信服务提供安全前提. (2) 建立基于信任根 DICE 的远程证明机制和访问控制机制实现安全认证和安全通信信道的建立. (3) 最终利用可信物联网设备为合法主机用户提供可信的安全存储服务, 在实现数据安全存储的同时, 兼顾隔离性和使用过程的灵活性. 实验结果表明, 本方案提供的安全存储服务具有较高的文件传输速率, 并具备较高的安全性, 可满足通用场景下的数据安全存储需求.

关键词: 物联网设备; 安全存储; 轻量级信任根; 可信启动; 远程证明

引用格式: 王辉,冯伟,秦宇.基于 DICE 的证明存储方案.计算机系统应用,2023,32(9):53–66. http://www.c-s-a.org.cn/1003-3254/9228.html

DICE-based Attestation and Storage Scheme

WANG Hui[1,2], FENG Wei[2], QIN Yu[2]

[1](University of Chinese Academy of Sciences, Beijing 100049, China)
[2](Trusted Computing and Information Assurance Laboratory, Institute of Software, Chinese Academy of Sciences, Beijing 100190, China)

Abstract: The continuous development of information technology and the popularization of intelligent terminal devices have led to the continuous growth of the total amount of global data storage, and the threats and challenges faced by data have increased with the prominence of their importance. However, currently, some computing and storage devices still lack data protection modules or have weak data protection capabilities. Existing data security storage technologies generally protect data through encryption, but data encryption and decryption operations, or data protection processes, are usually performed on the applied devices, resulting in threats to the security of stored data when the applied devices are subjected to various attacks. In response to the above issues, this study proposes a DICE-based Internet of Things (IoT) device attestation storage scheme, which utilizes trusted IoT devices built based on the lightweight root of trust DICE to provide secure storage services for general-purpose computing devices (collectively referred to as hosts), moves data encryption and decryption operations to trusted IoT devices, and eliminates threats to stored data caused by risks such as host memory attacks. This study mainly includes the following three aspects: (1) building a trusted IoT device by using

① 基金项目: 国家重点研发计划 (2022YFB4501500, 2022YFB4501501, 2020YFE0200600)
收稿时间: 2023-02-16; 修改时间: 2023-03-20; 采用时间: 2023-04-07; csa 在线出版时间: 2023-07-14
CNKI 网络首发时间: 2023-07-17

the root of trust DICE to provide a security prerequisite for providing trusted services; (2) establishing a DICE-based remote attestation mechanism and access control mechanism to achieve secure authentication and establish a secure communication channel; (3) using the trusted IoT device to provide trusted and secure storage services for legitimate host users, which achieves secure data storage and takes into account isolation and flexibility in the use process. The experimental results show that the secure storage service provided by this scheme has a high file transfer rate and high security, which can meet the requirements for secure data storage in general scenarios.

Key words: Internet of Things (IoT) device; secure storage; lightweight root of trust; trusted boot; remote attestation

1 引言

随着互联网技术和云计算、人工智能等新兴技术的持续发展, 以及智能终端等计算设备数量规模的不断扩大, 现如今全球的数据存储总量正以惊人的速度不断增长. 国际数据公司 IDC 预计在 2025 年全球的物联网设备或类似设备的数量规模会达到 416 亿, 并产生 79.4 ZB 的数据[1]. 而随着网络信息技术的不断发展, 数据资源在各行各业中地位的重要性不断凸显, 一些网络攻击者正在将数据资源作为他们破坏和窃取的对象. 而数据安全不但关乎国家、企事业单位的机密信息, 同时也与个人隐私紧密相关, 因此提高数据在全生命周期中的安全性成为亟需解决的挑战. 其中数据存储阶段是数据生命周期的关键阶段, 因此提高数据在存储阶段的安全性尤为重要. 在通用计算设备上, 实现数据的安全存储主要依赖于磁盘加密技术, 它包括基于硬件和基于软件两种类型, 基于硬件的磁盘加密技术依赖于特定的硬件驱动器, 例如自加密驱动器 SED[2]; 而基于软件的磁盘加密技术以运行在内核层的应用程序为基础, 例如 eCryptfs. 虽然以上两种磁盘加密技术都可以在一定程度上提高数据在存储期间的安全, 但是都存在一定的缺陷, 例如基于硬件的磁盘加密技术需要依赖特定的硬件, 而部分基于软件的磁盘加密技术由于缺乏隔离性, 在遭受主机内存攻击时无法保证加载至主机内存中密钥的安全. 同时例如 Windows 提供的驱动器加密工具 BitLocker[3] 在使用过程中会将加密密钥释放到内存中, 在主机内存遭受攻击时可能会导致密钥泄漏进而影响存储数据的安全. 针对以上问题, 可以考虑借助外部设备为存在数据安全存储需求的计算设备提供安全存储服务, 在不增加原有计算设备硬件成本的同时兼顾与外部设备之间的隔离性, 进而提高存储数据的安全性. 而小型物联网设备具有便携、成本较低且功能较为全面的优势, 因此使用物联网设备来提供安全存储服务具备较高的可行性.

物联网设备通常利用传感器等硬件对真实物理环境进行探测, 并利用传统互联网技术与其他计算设备进行数据传输. 物联网设备通常由感知层、网络层和应用层[4] 构成. 作为互联网技术和硬件技术结合发展的产物, 物联网设备同样面临着来自传统网络安全风险的威胁, 例如中间人攻击等网络攻击. 同时随着物联网技术与云计算、人工智能等新兴技术的融合发展, 物联网设备在硬件、软件和数据 3 个方面面临着愈加严峻的风险挑战.

因此, 为了在物联网设备上构建安全存储服务, 同时防止针对物联网设备发起的攻击, 有必要基于可信计算技术构建可信物联网设备, 使其具备平台身份和平台状态证明的能力. 而基于信任根的可信计算技术是满足物联网设备在信任建立、授权访问控制和数据机密性、完整性保护[5] 等方面安全需求的常用技术. 普通计算设备通常利用基于硬件信任根的可信计算技术实现自身对攻击的主动防御. 常见的传统硬件信任根有 TPM (trusted platform module)/TCM (trusted cryptography module)[6]. 但由于部分物联网设备与个人电脑、服务器等普通计算设备相比, 在内存、计算能力等资源以及能源储备方面存在较大的局限性[7], 因此无法直接使用传统硬件信任根 TPM/TCM 实现可信启动[8]、远程证明[9] 以及数据的安全存储.

针对以上问题, 本文提出一种基于信任根 DICE (device identifier composition engine)[10] 的证明存储方案, 利用轻量级信任根 DICE 实现物联网设备的可信启动以创建可信计算环境, 并基于该可信物联网设备向合法主机提供可信的安全存储服务, 其中 DICE 是国际可信计算组织提出的一种可作为轻量级信任根的安全架构标准. 该方案由可信启动、远程证明和安全存储部分组成, 首先将轻量级信任根 DICE 作为物

联网设备的信任锚点并建立信任链实现可信启动, 然后基于可信启动过程中派生的复合设备标识符 *CDI* (compound device identifier) 生成用于物联网设备平台身份和平台状态证明的证明密钥以实现自身平台身份和平台状态的证明, 并验证主机身份的合法性. 随后使用磁盘加密技术将物联网设备连接的外置移动存储设备制作为加密分区作为持久化存储区域, 在完成远程证明的基础上结合 FTP 技术向合法主机提供安全存储服务. 本文的主要贡献如下.

(1) 利用基于信任根 DICE 构建的可信物联网设备为主机提供安全存储服务, 将存储数据的加解密操作隔离至可信物联网设备上执行, 解决了磁盘加密技术等安全存储技术所存在的因主机遭受内存攻击等风险而威胁存储数据安全的问题.

(2) 建立基于信任根 DICE 的物联网设备远程证明机制和加密分区的访问控制机制, 确保提供安全存储服务的物联网设备、请求服务的主机和用户处于可信状态.

(3) 在原有 DICE 的固件分层度量引导基础上设计了一种树形度量引导模型以实现对物联网设备完整固件层或固件层部分组件的选择性度量, 在实现物联网设备可信启动的同时降低在固件层组件耦合度较低的情况下不必要的度量引导开销.

本文第 2 节介绍国内外在可信启动、远程证明以及安全存储方面的相关工作. 第 3 节介绍本方案的系统模型、威胁模型和安全假设. 第 4 节介绍本方案各模块的功能与设计. 第 5 节介绍本方案的实验过程并对实验结果进行性能评估和安全性分析. 第 6 节对全文进行总结并对未来工作进行展望.

2 相关工作

2.1 可信启动

可信启动[8]是一种在计算设备通电重启后通过利用信任根构建信任链的方式实现对计算设备操作系统、固件等组件度量引导的技术. 其中信任根和信任链是实现可信启动的关键, 信任根是固化在计算设备内部的最小安全功能组件, 具有防物理篡改的优势[11], 比较常见的硬件信任根是安全芯片. 国际可信计算组织发布的 TPM 和中国发布的 TCM 是两种主流的安全芯片标准.

2.1.1 基于传统信任根 TPM 的可信启动

如图 1 所示, 当计算设备通电重启后, 处于可信引导块中的 CRTM 最先开始运行, 在完成自检之后对可信引导块的剩余部分进行度量, 随后将引导控制权转交给可信引导块. 随后以同样的方式完成从可信引导块到操作系统加载程序、再到操作系统直至系统应用的引导过程[12]. 在此阶段内, TPM 对计算设备各阶段运行代码进行计算度量, 同时利用平台状态寄存器进行扩展度量, 并将最后的度量值记录在度量日志中.

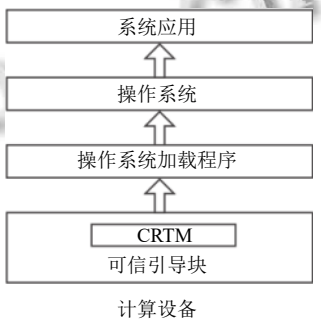

图 1 基于 TPM 的可信启动

2.1.2 基于轻量级信任根 DICE 的可信启动

针对部分资源受限、无法内置传统信任根 TPM/TCM 的计算设备, 国际可信计算组织提出了一种可作为轻量级信任根的安全架构标准 DICE, 用于实现此类计算设备的度量引导[13]. 实现 DICE 的最低硬件需求仅包括以下 3 个部分: (1) 作为计算设备全部信任基础和设备身份密钥的唯一设备秘密 *UDS* (unique device secret); (2) 计算设备重启后最先运行的引导代码; (3) 用于阻止除引导代码外的固件访问 *UDS* 的锁定机制[10].

基于 DICE 的度量引导方案中固件分为一至多层, 并依次获得控制权[13], 假设有 n 层, 为L_0, L_1, L_2,$\cdots$, L_{n-1}. 各层固件按照顺序进行逐级度量引导, 如图 2 所示. 当设备通电重启后, DICE 短暂访问 *UDS* 并利用单向函数基于设备身份密钥 *UDS* 和固件层L_0的度量值为固件层L_0生成一个称为$CDI(L_0)$的复合设备标识符, 除最后一层外每层固件都会为下层固件生成属于该下层固件自身的复合设备标识符. 假设当前运行固件层为L_t, L_t在获得L_{t-1}传递的控制权和复合设备标识符$CDI(L_t)$之后, 在实现对L_{t+1}层固件的度量后利用单向函数基于自身复合设备标识符$CDI(L_t)$和L_{t+1}的固件度量值为L_{t+1}生成复合设备标识符$CDI(L_{t+1})$, 即:

$$CDI(L_{t+1}) = OWF(CDI(L_t), Hash(L_{t+1})) \quad (1)$$

随后将引导控制权和$CDI(L_{t+1})$一并转交给L_{t+1}层固件. 该方案中每层固件的复合设备标识符都可用于派生属于该层固件的数据密封密钥和远程证明密钥.

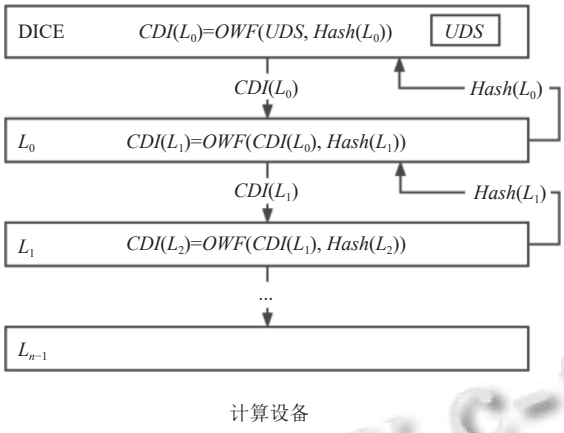

图 2 基于 DICE 的度量引导

2.2 远程证明

远程证明是指远程验证方对可疑计算设备的平台身份、平台状态进行验证的技术, 包括平台身份证明和平台状态证明. 从实现方式角度区分, 远程证明可以分为基于软件、基于硬件以及基于软硬件协同这 3 种方式, 基于软件的方式仅依赖于证明代码, 以 SWATT[14]、Reflection[15] 为代表. 基于硬件的方式需要借助硬件来实现远程证明过程, 基于硬件信任根 TPM/TCM 和 Intel SGX[16] 的远程证明是基于硬件方式的代表性方案. 而基于软硬件协同的远程证明是指利用具有最小硬件安全特征的硬件组件实现远程证明的方式, 以 TyTan[17]、TrustLite[18] 为代表.

2.2.1 基于软件的远程证明

基于软件的远程证明是指对目标计算设备内存中的数据、代码等进行验证的过程. Reflection[15] 是最早的软件证明方案, 它的基本思想是构造两个包含覆盖完整内存空间地址范围的随机挑战, 并且所包含的地址范围互相重叠, 以实现对目标设备完整内存地址空间的验证. 但由于缺乏硬件模块的保护, 此类证明方案容易遭受内存压缩、内存复制等攻击.

针对以上软件证明的风险挑战, 目前的应对方案主要包括以下 3 种[11]: (1) 基于严格证明时间的软件证明: 利用伪随机遍历等手段防止攻击者影响证明过程并对远程验证方造成欺骗, SWATT[14] 是此类方案的典型代表, 但由于对证明时间的强烈依赖, 因此需要保证自身算法始终处于最优状态. (2) 基于空闲内存填充的软件证明: 利用伪随机噪声对设备空闲程序内存进行填充以防止敌手对空闲内存空间进行非法占用, 在分布式场景下提出的方案 [19] 是该类方案的典型代表. (3) 基于随机证明函数的软件证明: 利用不可预测的随机构造证明函数代替固定证明函数, PIV[20] 是此类方案的典型代表, 该方案会在证明时通过生成随机哈希函数的方式提高证明过程的安全性.

2.2.2 基于硬件的远程证明

基于硬件的远程证明是指利用硬件信任根、支持可信执行环境[21] 的处理器或者其他硬件技术实现远程证明的一种方式. TPM 是最常用于远程证明的硬件信任根, 基于 TPM 的平台状态证明过程一般是首先计算当前设备的平台状态度量值, 再利用自身独有的签名私钥基于该度量值生成签名, 然后将平台状态度量值和签名作为可信性证据一起发送给验证方, 验证方收到后对平台状态度量值和签名进行验证. 当需要在远程证明过程中保持证明方身份的匿名性时, 可以利用直接匿名证明协议 DAA (direct anonymous attestation)[22] 实现远程证明过程.

英特尔公司推出的 Intel SGX 是一款支持可信执行环境的商用处理器, 它支持本地证明和远程证明两种证明方式[23], 其中本地证明是远程证明的基础. 本地证明是指两个处于同一计算平台上的 enclave 之间进行双向验证的过程, 而远程证明在不同计算平台之间进行, 由应用程序、应用程序 enclave 和负责生成远程证明报告的 Quoting enclave 协作完成, enclave 是存放运行中敏感代码和数据的安全容器[16].

2.2.3 基于软硬件协同的远程证明

基于软硬件协同的远程证明是一种利用最小硬件安全特征实现远程证明的方式, 代表性工作有 TyTan、TrustLite. TyTan[17] 是首个针对嵌入式系统提出的安全架构, 它通过计算任务二进制代码的哈希值作为任务本地证明的身份标识和证明依据, 并利用远程证明密钥生成 MAC (message authentication code) 作为远程证明的证据, 该远程证明密钥基于平台密钥生成. 为了保证平台密钥的安全性与远程证明报告的可信性, TyTan 通过内存保护单元实现对平台密钥的保护.

国际可信计算组织提出了一种基于 DICE 的远程证明方案[24], 该方案通过验证计算设备生成的 MAC 是

否正确, 从而判断出计算设备用于生成 MAC 的密钥是否合法, 由于生成 MAC 的远程证明密钥是在 *UDS* 和设备固件度量值基础上生成的, 因此远程证明密钥可以正确反映该设备平台身份和平台状态的可信性.

2.3 安全存储

数据安全是计算设备安全体系的关键组成部分, 包括数据机密性、数据完整性和数据可用性. 下面依次对常见数据安全存储技术 BitLocker 和 IPFS 进行介绍.

微软的 BitLocker[3] 是 Windows 操作系统提供的一种驱动器加密工具, 可用于对系统分区、磁盘分区以及可移动存储介质中的数据进行加密保护, 密钥 FVEK 和密钥 VMK 是数据保护过程中的关键密钥, 密钥 FVEK 用于对数据直接进行加密保护, 而密钥 VMK 用于对密钥 FVEK 进行加密保护. BitLocker 可以实现对存储数据的有效保护, 但仍存在一定的安全风险, 主要来源于以下两个方面: 一方面是存在敌手通过暴力破解获得 VMK 密钥的风险; 另一方面是存在释放到内存中的密钥被窃取导致加密数据泄露的风险.

IPFS (Intel protection file system)[25] 是 Intel SGX 提供的具备安全特性的用户级文件系统, 有数据安全存储需求的应用程序可以通过创建 enclave 的方式来调用 IPFS 提供的接口, 对数据进行加密存储, 实现数据的机密性保护. 但由于 IPFS 在文件读写时仍需要借助主机系统的不可信库, 因此无法实现对新鲜数据和持久化数据的完整性保护, 所以 IPFS 只能实现对加密数据篡改行为的事后检测, 但无法保证加密数据不受到恶意篡改.

3 系统模型与安全假设

本文面向通用计算设备 (称为主机) 设计了一种基于可信物联网设备的证明存储方案, 并基于轻量级信任根 DICE 构建可信物联网设备, 以下是对该证明存储方案系统模型、威胁模型的介绍与安全假设的说明.

3.1 系统模型

该方案的系统模型由可信物联网设备和主机两部分组成, 如图 3 所示. 可信物联网设备由基于 DICE 的设备端可信启动模块、设备端远程证明模块和设备端安全存储模块构成. 基于 DICE 的设备端可信启动模块用于实现物联网设备固件的可信启动和复合设备标识符 *CDI*、远程证明密钥的生成. 设备端远程证明模块用于自身平台状态和平台身份的证明, 以及对主机的平台身份进行验证. 设备端安全存储模块由 FTP 服务端和加密分区管理子模块构成, 用于加密分区的制作、管理和实现对加密分区的访问控制, 以及实现与 FTP 客户端之间的数据传输. 而主机由主机端远程证明模块、主机端安全存储模块构成, 主机端远程证明模块用于验证物联网设备的平台状态、平台身份以及实现自身的平台身份证明. 主机端安全存储模块由 FTP 客户端和主机端加密分区管理子模块构成, 用于实现与 FTP 服务端之间的数据传输和方便主机用户对加密分区进行管理, 可信物联网设备和主机之间通过本地网络进行数据传输.

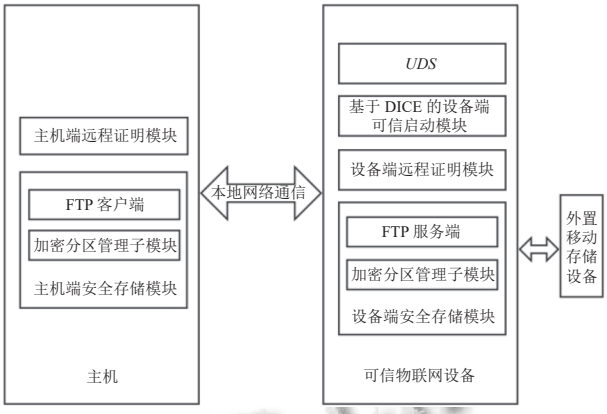

图 3 系统模型

当物联网设备通电重启后, 物联网设备的基于 DICE 的设备端可信启动模块在操作系统可信启动的基础上, 基于轻量级信任根 DICE 对设备完整固件层或固件层部分组件进行度量引导实现固件的可信启动和复合设备标识符 *CDI* 的生成, 并基于设备固件的复合设备标识符 *CDI* 派生物联网设备的远程证明密钥. 在完成物联网设备固件的可信启动后, 设备端远程证明模块利用可信启动阶段生成的远程证明密钥实现物联网设备自身的平台身份、平台状态证明, 同时对主机的平台身份进行验证. 在完成远程证明之后, 通过身份验证的合法主机用户可以向可信物联网设备请求安全存储服务, 通过主机端安全存储模块对加密分区中的文件进行上传、下载和显示操作.

3.2 威胁模型

(1) 攻击者具备在物联网设备启动期间对设备固

件进行攻击的能力, 即攻击者可以对设备固件代码进行恶意篡改或植入恶意代码.

(2) 攻击者在物联网设备与主机进行远程证明的过程中试图冒充合法物联网设备或主机.

(3) 外置移动存储设备存在被攻击者非法窃取或处于丢失等非安全状态下的风险可能.

3.3 安全假设

(1) 针对主机的安全假设

本方案仅考虑主机大规模存储数据的安全性, 主机数据在使用阶段的安全性不在本方案的考虑范围内, 数据在使用阶段的安全可通过可信执行环境实现.

(2) 针对可信物联网设备的安全假设

1) 作为物联网设备身份密钥的 *UDS* 和设备端可信启动模块是受物理防篡改保护的, 是固化在物联网设备中的绝对信任, 攻击者无法对其进行篡改和破坏.

2) 仅设备端可信启动模块在通电重启后可访问 *UDS*, 攻击者无法访问物联网设备身份密钥 *UDS* 和可信启动模块的代码.

3) 仅自身可访问可信启动过程中基于复合设备标识符 *CDI* 派生的密钥, 攻击者无法访问、篡改和破坏.

4) 向合法主机提供安全存储服务时, 其自身的操作系统内核始终是安全的, 攻击者无法对其进行劫持和破坏, 并且不考虑针对 DICE 的 TOCTOU 攻击.

5) 攻击者的攻击能力仅限于发动网络攻击, 无法接触物联网设备, 不考虑针对设备发起的窥探、破坏等物理攻击, 但是该证明存储方案无法有效抵御 DDOS 攻击和侧信道攻击.

(3) 针对外置移动存储设备的安全假设

仅可信物联网设备能够打开和关闭外置移动存储设备的加密分区.

3.4 符号说明

表 1 列出本方案所用的标识符.

4 方案设计

4.1 基于 DICE 的物联网设备可信启动

本方案在物联网设备操作系统可信启动的基础上, 基于轻量级信任根 DICE 实现物联网设备固件的可信启动. 传统基于 DICE 的计算设备度量引导方案是将固件划分为多层, 然后按照固件分层顺序对固件进行线性的逐级度量引导并建立信任链, 同时在信任链建立过程中为各层固件派生复合设备标识符 *CDI*, 并利用密钥派生函数基于复合设备标识符 *CDI* 派生用于远程证明和对数据加密保护的密钥. 但是在计算设备的单一固件层过于庞大、构成组件之间耦合度较低的情况下, 该方案会增加不必要的计算开销, 存在效率较低的问题.

针对该问题, 本方案基于 DICE 提出一种实现对计算设备完整固件层或固件层部分组件进行选择性度量的树形度量引导模型, 该树形度量引导模型将前一引导阶段作为根结点, 将当前固件层的各组件作为子节点, 根据实际的固件组件使用情况实现对部分固件组件的扩展度量或完整固件层的度量. 如图 4 所示, 以单个固件层为例, 假设该层固件有 n 个构成组件, 分别与树中各个子节点相对应, 即 $f_1, f_2, f_3, \cdots, f_n$.

表 1 标识符与函数标识

标识符与函数标识	说明
DICE	物联网设备信任根
UDS	物联网设备身份密钥
FIRMWARE	物联网设备固件
f_i	设备固件的第i个组件
CDI	复合设备标识符
CDI_{Lt}	第t层固件的复合设备标识符
user_name	主机用户登录名
user_pass	主机用户登录口令
ep_pass	加密分区访问密码
Dev_sk、*Dev_pk*	物联网设备非对称密钥对
Alias Key	物联网设备对称密钥
H_sk、*H_pk*	主机非对称密钥对
H_k	主机对称密钥
KEK	密钥加密密钥
DEK	加密分区关联密钥
DEK.encfile	加密分区关联密钥的密文文件
D_sig、*H_sig*	物联网设备、主机生成的签名
D_hmac、*H_hmac*	物联网设备、主机生成的*HMAC*
D_r、*H_r*	物联网设备、主机的签名验证结果
*H_n*1、*H_n*2	主机生成的随机消息
*D_n*1、*D_n*2	物联网设备生成的随机消息
H_k' 、*Alias Key'*	主机、物联网设备合法对称密钥
D _hmac'、*H_hmac'*	物联网设备、主机应生成的标准*HMAC*
Hash(code)	code的哈希值
OWF(*k*, *Hash*(code))	单向函数基于密钥*k*和code哈希值生成密钥
Signature(*m*, *sk*)	利用私钥*sk*对消息*m*签名
Verify(*m*, *pk*, *sig*)	利用公钥*pk*验证消息*m*的签名*sig*
HMAC(key, *Hash*(code))	基于密钥key和code哈希值生成*HMAC*

若需要运行完整设备固件层, 就计算该设备固件层的完整度量值, 并利用单向函数基于物联网设备的身份密钥 *UDS* 和完整固件层度量值生成该固件层的

复合设备标识符 *CDI*, 如图 5 所示. 当仅需运行 f_1 节点对应的固件组件时就对该组件的代码进行度量, 并利用单向函数基于 *UDS* 和该固件组件的度量值生成标识符, 作为当前该固件层的复合设备标识符 *CDI*, 而无需对该组件所属的完整固件层进行度量, 可以有效减少计算开销、提高度量引导效率, 如图 6 所示. 此时可以利用密钥派生函数基于复合设备标识符 *CDI* 生成远程证明密钥, 实现设备的平台身份证明和设备固件层的完整性证明.

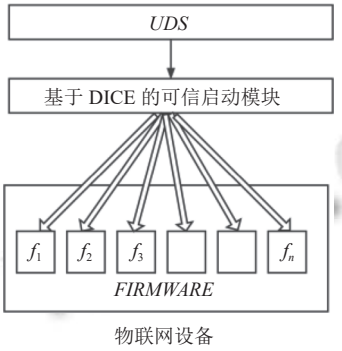

图 4 基于 DICE 的树形度量引导模型

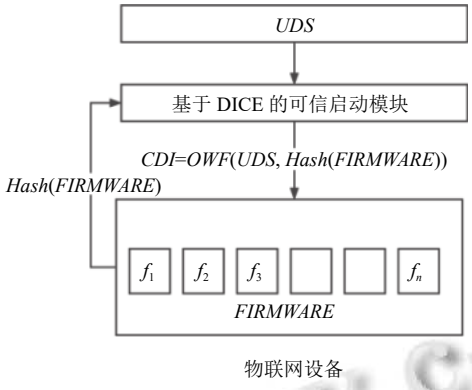

图 5 完整设备固件层度量引导

4.2 基于 DICE 的远程证明

在基于轻量级信任根 DICE 完成物联网设备的可信启动并创建可信计算环境后, 物联网设备需要向主机证明自身平台身份和平台状态的可信性, 并且对主机的身份进行验证. 由于物联网设备在可信启动阶段生成的复合设备标识符 *CDI* 是基于设备身份密钥 *UDS* 和设备完整固件层或固件层部分组件的度量值生成的, 因此基于 *CDI* 生成的远程证明密钥可用于同时实现物联网设备的平台状态证明和平台身份证明. 与可信启动阶段的树形度量引导模型相对应, 物联网设备的平台状态证明对象可以是完整设备固件层, 也可以是设备固件层的部分组件. 前一种将可信启动阶段中基于设备身份密钥 *UDS* 和完整设备固件层度量值生成的复合设备标识符 *CDI* 作为派生远程证明密钥的种子, 而后一种证明过程中的复合设备标识符 *CDI* 是基于设备身份密钥 *UDS* 和设备固件层部分组件的度量值生成的. 证明过程可以利用数字签名和 *HMAC* 两种方式实现, 主机用于生成签名的非对称密钥对 *H_sk*、*H_pk* 和生成 *HMAC* 的对称密钥 *H_k* 是向可信第三方请求得到的.

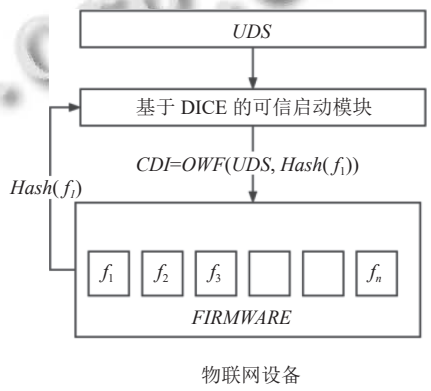

图 6 固件层部分组件度量引导

基于数字签名的物联网设备平台状态证明以及与主机之间的双向身份认证过程如图 7 所示, 证明开始后物联网设备向主机发送随机挑战消息 *D_n*1, 主机收到后利用私钥 *H_sk* 基于 *D_n*1 生成签名 *H_sig*, 即:

$$H\_sig = Signature(D\_n1, H\_sk) \tag{2}$$

随后主机生成随机消息 *H_n*1, 并将其和签名 *H_sig* 作为响应消息返回给物联网设备, 物联网设备收到签名 *H_sig* 后使用公钥 *H_pk* 对其进行验证, 即:

$$H\_r = Verify(D\_n1, H\_pk, H\_sig) \tag{3}$$

随后物联网设备利用自身拥有的签名私钥 *Dev_sk* 基于随机消息 *H_n*1 生成签名 *D_sig* 并返回给主机, 即:

$$D\_sig = Signature(H\_n1, Dev\_sk) \tag{4}$$

主机收到物联网设备的签名后使用合法公钥 *Dev_pk* 对其进行验证, 即:

$$D\_r = Verify(H\_n1, Dev\_pk, D\_sig) \tag{5}$$

由于物联网设备的签名私钥与 *CDI* 相关联, 而 *CDI* 是基于 *UDS* 和固件度量值生成的. 若物联网设备签名的验证结果是正确的, 说明物联网设备的平台身份和当前平台状态都是可信的, 否则至少有一个不可

信; 若主机所生成签名的验证结果也是正确的则表明主机的平台身份也是可信的, 反之不可信.

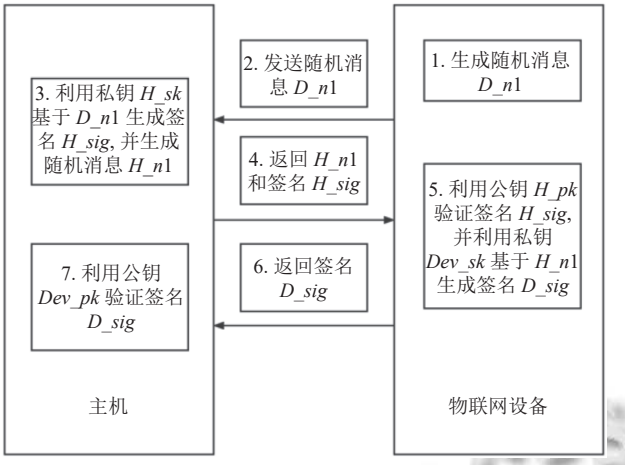

图 7 基于数字签名的远程证明

基于 *HMAC* 的物联网设备平台状态证明和与主机之间的双向身份认证过程如图 8 所示, 物联网设备生成随机消息 *D_n*2 作为远程证明挑战发送给主机, 主机收到证明挑战后利用对称密钥 *H_k* 基于 *D_n*2 生成 *HMAC*, 即:

$$H\_hmac = HMAC(H\_k, D\_n2) \tag{6}$$

随后主机生成随机消息 *H_n*2 作为远程证明挑战和 *H_hmac* 一起作为响应消息返回给物联网设备. 而物联网设备收到响应消息后先利用自身拥有的合法密钥 *H_k*′基于发送的随机挑战消息 *D_n*2 生成标准 *HMAC* 即 *H_hmac*′与 *H_hmac* 进行比对, 即:

$$H\_hmac' = HMAC(H\_k', D\_n2) \tag{7}$$

随后物联网设备利用对称密钥 *Alias Key* 基于随机消息 *H_n*2 生成 *HMAC* 即 *D_hmac* 作为响应消息返回给主机, 即:

$$D\_hmac = HMAC(Alias\ Key, H\_n2) \tag{8}$$

主机收到后利用合法密钥 *Alias Key*′基于随机消息*H_n*2 生成标准 *HMAC* 即 *D_hmac*′作为标准响应消息, 即:

$$D\_hmac' = HMAC(Alias\ Key', H\_n2) \tag{9}$$

若 *D_hmac* 与 *D_hmac*′相同则代表物联网设备的身份和平台状态都是可信的, 若不同则说明物联网设备的平台身份和平台状态至少有一个不可信; 若 *H_hmac* 和 *H_hmac*′相同说明主机的平台身份是可信的, 反之不可信.

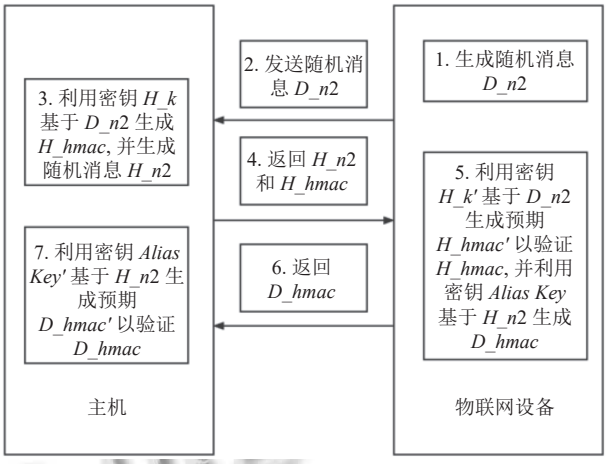

图 8 基于*HMAC*的远程证明

4.3 基于 FTP 的安全存储

安全存储的主要工作是加密分区的制作、管理和 FTP 服务端、FTP 客户端的改造与部署. 当可信物联网设备基于轻量级信任根 DICE 完成自身平台状态和平台身份的证明, 并对主机的平台身份进行成功验证后就可以向合法主机用户提供可信的安全存储服务. 物联网设备的设备端安全存储模块由 FTP 服务端和加密分区管理子模块组成, 加密分区管理子模块负责加密分区制作和管理, 将与物联网设备连接的外置移动存储设备制作为加密分区, 作为数据的持久化加密存储区域. 同时生成密钥加密密钥 KEK 对与加密分区关联的随机密钥 DEK 进行加密保护, 即生成密钥 DEK 对应的密文文件 DEK.encfile, 仅通过身份验证的主机合法用户可以凭借正确的加密分区密码请求可信物联网设备使用密钥 KEK 对密钥 DEK 的密文文件 DEK.encfile 进行解密获得该密钥的明文形式, 从而进一步获得加密分区的访问权限. 密钥 DEK 长期以密文形式存储在可信物联网设备中并受到保护, 并仅在可信物联网设备为拥有正确加密分区密码的合法主机用户提供安全存储服务时才会被解密, 有效实现了加密密钥与主机的分离, 避免了因主机遭受主机内存攻击带来的密钥泄漏等安全风险, 并且仅拥有合法密钥 KEK 的可信物联网设备才能对密钥 DEK 的密文文件进行解密进而打开加密分区, 更大程度上提高了存储数据的安全性.

物联网设备与主机之间利用 FTP 技术实现数据传输, 为了方便合法主机用户对加密分区进行操作并提高物联网设备加密分区的安全性, 对分别部署在物联网设备和主机上的 FTP 服务端、FTP 客户端进行改

造, 在原有 FTP 功能基础上增加访问控制机制和加密分区管理机制. 在 FTP 服务端一侧保存合法主机用户对应的登录口令哈希值和正确加密分区密码的哈希值, 分别作为合法的主机用户身份凭证和正确的加密分区访问凭证. 并在 FTP 服务端一侧增加加密分区管理功能接口, 实现加密分区的打开、关闭等管理操作, 同时将 FTP 服务端的根目录路径与加密分区进行绑定, 并且可以考虑扩展物联网设备的设备端远程证明模块功能至 FTP 服务端实现物联网设备的平台状态和身份证明以及对主机的身份验证. 而在 FTP 客户端一侧增加哈希值计算接口, 用于计算主机用户登录口令 user_pass 的哈希值 *Hash*(user_pass) 以及加密分区密码 ep_pass 的哈希值 *Hash*(ep_pass), 并且增加加密分区管理功能方便主机用户打开和关闭物联网设备的加密分区, 仅通过身份认证的合法主机用户可以凭借正确的加密分区密码请求可信物联网设备打开和关闭加密分区. 与 FTP 服务端一样可以考虑扩展主机端远程证明模块的功能至 FTP 客户端, 实现对物联网设备平台状态和平台身份的验证, 以及主机自身的平台身份证明.

如图 9 所示, 当主机用户向物联网设备请求安全存储服务时, 需要先凭借正确的登录名和登录口令证明自身身份的合法性, 随后再凭借合法加密分区密码 ep_pass 请求物联网设备使用密钥加密密钥 KEK 对与加密分区关联的随机密钥 DEK 的密文文件 DEK.encfile 进行解密, 进一步获得加密分区的访问权限即打开加密分区, 之后合法主机用户就可以对加密分区中的文件进行显示、上传和下载操作. 当合法主机用户使用完加密分区后可以再次凭借正确的加密分区密码 ep_pass 请求物联网设备关闭加密分区, 随后可信物联网设备就会为拥有正确加密分区密码 ep_pass 的合法主机用户关闭加密分区, 并删除与加密分区关联的随机密钥 DEK 的明文文件以防止加密分区中的数据被非法窃取.

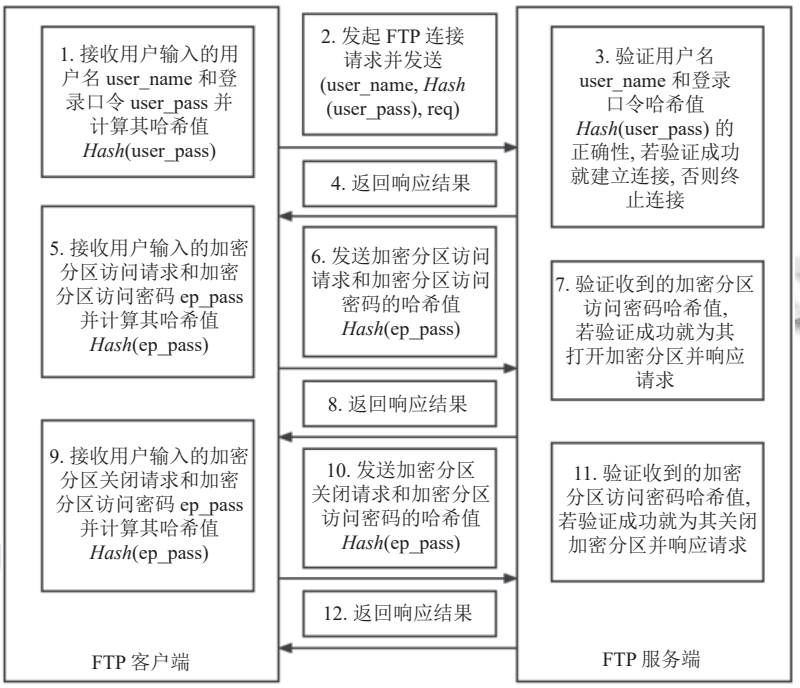

图 9 主机用户请求安全存储服务的流程

5 实验与评估

本节先介绍了本方案依赖的硬件设备, 然后描述了本方案的设计与实现方式, 最后对各模块进行了实验测试, 并在实验结果基础上对性能表现进行分析, 同时对本方案的安全性进行分析.

5.1 硬件平台

本文选择树莓派开发板作为物联网设备, 设备型号为 Raspberry Pi 4 Model B, 它所搭载的操作系统为 Debian GUN/Linux 11, 硬盘为 16 GB 的 Micro SD 卡, 并且自身携带多个 USB 接口, 可连接用于制作加密分

区的外部移动存储设备, 同时还提供用于连接网线的以太网接口和无线网络通信模块, 可方便与主机之间进行数据通信, 并且它的体积较小, 具有易于携带和价格较低的优势. 主机部分则选择部署在个人电脑的虚拟机中, 个人电脑的设备型号为 MacBook Pro, 该设备搭载主频为 2 GHz 的四核 Intel Core i5 处理器, 内存空间大小为 16 GB, 虚拟机的操作系统版本为 Ubuntu 16.04. 外置移动存储设备是 SanDisk 品牌的 USB 闪存驱动器, 其存储空间大小为 32 GB. 主机和物联网设备之间通过本地网络进行通信, 图 10 是主机和物联网设备树莓派开发板之间通过本地网络进行连接的实物图.

图 10 主机与物联网设备的本地网络连接实物图

5.2 方案实现

5.2.1 物联网设备可信启动

本方案在物联网设备操作系统可信启动的基础上按照微软和国际可信计算组织制定的 DICE 相关规范, 提前向可信第三方请求长度为 256 位的随机秘密值作为物联网设备身份密钥, 即唯一设备秘密 *UDS*. 当物联网设备通电重启后, 按照树形度量引导模型建立信任链, 实现物联网设备固件的可信启动. 本方案将物联网设备的设备端远程证明模块和设备端安全存储模块作为完整设备固件层, 在实验过程中分别对完整固件层和设备端安全存储模块进行了可信启动测试.

在实现完整固件层可信启动过程中利用 HMAC-SHA256 算法基于物联网设备的设备身份密钥 *UDS* 和完整固件层的度量值生成复合设备标识符 *CDI*, 固件层度量值是使用 SHA-256 算法计算得到的, 并利用 Python 函数库提供的密钥导出函数基于复合设备标识符 *CDI* 进一步生成用于物联网设备远程证明的证明密钥, 同时将生成的 *CDI* 和远程证明密钥分别写入日志文件和密钥文件中, 其中复合设备标识符 *CDI* 作为物联网设备平台状态证明的证据. 若生成的 *CDI* 和预期结果一致, 说明在通电重启后物联网设备的身份和平台状态都是可信的, 否则则说明至少有一个不可信. 而实现设备端安全存储模块可信启动时, 利用 HMAC-SHA256 算法基于物联网设备的设备身份密钥 *UDS* 和设备端安全存储模块的度量值生成复合设备标识符 *CDI*, 并基于其派生远程证明密钥, 随后将生成的 *CDI* 和远程证明密钥分别写入日志文件和密钥文件中.

5.2.2 远程证明模块

本方案为物联网设备的平台状态证明和与主机之间的双向身份认证过程提供了数字签名和 *HMAC* 两种实现方式. 数字签名算法选择国产椭圆曲线密码算法 SM2-256 和国际上的 RSA-2048 密码算法进行性能对比, 而为了对比 SM2-256 算法和 RSA-2048 算法的性能表现, 物联网设备和主机所使用的非对称密钥对是向可信第三方请求获得的. 同时基于 HMAC-SHA256 的远程证明选择长度为 256 位的对称密钥, 物联网设备使用的对称密钥是可信启动过程中利用 Python 函数库提供的密钥导出函数基于复合设备标识符 *CDI* 派生的, 而主机所使用的对称密钥是向可信第三方请求获得的. 同时本节对物联网设备的平台状态证明和与主机之间的双向身份认证进行了单独实现, 并且在物联网设备的平台状态证明过程中将可信启动过程中生成的复合设备标识符 *CDI* 作为平台状态证据.

5.2.3 安全存储模块

本方案选择 SanDisk 品牌且存储空间大小为 32 GB 的 USB 闪存驱动器作为物联网设备连接的外置移动存储设备, 并利用物联网设备搭载的 Linux 操作系统所提供的磁盘分区加密工具 Cryptsetup 将其制作为加密分区. 同时在物联网设备上移植国密 SM3 哈希算法、SM4 对称加密算法, 并生成对称密钥 KEK 作为密钥加密密钥, 利用对称密钥 KEK 和 SM4 对称加密算法实现对加密分区关联密钥 DEK 的加密保护, 同时利用基于 Python 的异步文件传输技术 aioftp 实现主机与物联网设备之间的文件传输. 为了实现对加密分区的访问控制并方便合法主机用户对加密分区中的文件进行操作, 在原有 aioftp 功能基础上对 FTP 服务端和 FTP 客户端进行改造, 在 FTP 服务端一侧保存两个相同合法用户登录名的用户信息, 第 1 个合法主机用户信息保存登录名、登录口令哈希值和空文件目录路径用于实现主机用户的身份验证, 第 2 个合法主机用户信息保存用户登录名、正确加密分区密码的哈希值以及加密分区的文件目录路径用于实现对加密分区的访

问权限验证, 并增加加密分区打开、关闭接口实现对加密分区的管理, 同时将 FTP 服务端的根目录路径与加密分区进行绑定.

而在 FTP 客户端一侧增加 SM3 哈希算法计算接口用于计算主机用户登录口令的哈希值 *Hash*(user_pass) 和加密分区密码的哈希值 *Hash*(ep_pass), 将其分别作为请求安全存储服务的身份凭证和加密分区的访问凭证与用户登录名发送给 FTP 服务端进行验证, 并增加加密分区管理接口方便合法主机用户请求可信物联网设备对加密分区进行打开、关闭操作. 在对 FTP 服务端、FTP 客户端改造完成之后将其分别部署在物联网设备和主机上.

5.3 实验结果与分析

本节对本方案所设计模块的相关性能进行了测试和分析, 包括物联网设备的可信启动性能、物联网设备的平台状态证明和与主机之间的双向身份认证性能以及安全存储模块的数据传输性能.

(1) 物联网设备可信启动的性能开销

物联网设备的可信启动阶段主要包括计算物联网设备固件层度量值、派生复合设备标识符 *CDI*、基于 *CDI* 派生远程证明密钥以及将复合设备标识符 *CDI* 和远程证明密钥分别写入日志文件和密钥文件 4 个步骤. 在不考虑实际使用过程中的远程证明需求, 本节分别测试了实现物联网设备完整固件层和固件层安全存储模块可信启动的时间开销, 测试数据如图 11、图 12 所示. 经过 10 次测试, 物联网设备完整固件层的平均可信启动时间为 320.256 ms. 而实现设备端安全存储模块可信启动的平均时间为 309.646 ms. 不考虑读取 *UDS* 以及对复合设备标识符 *CDI* 和远程证明密钥进行写操作的时间, 可信启动的主要时间开销来自于计算固件度量值、派生复合设备标识符 *CDI* 和远程证明密钥操作.

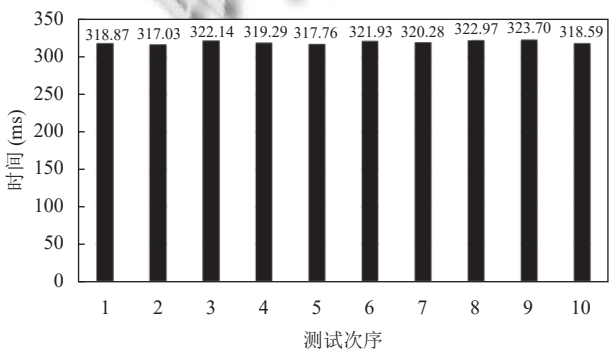

图 11 完整固件层可信启动时间

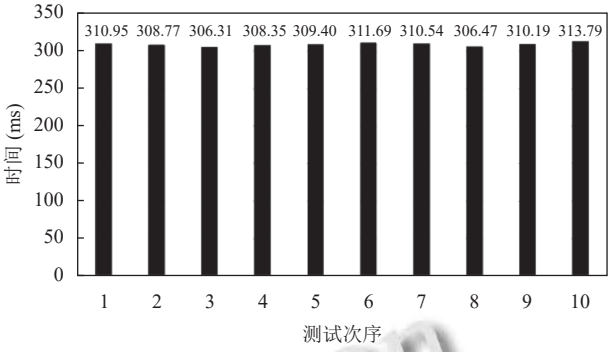

图 12 安全存储模块可信启动时间

(2) 远程证明的性能开销

远程证明包括物联网设备的平台状态证明和与主机之间的双向身份认证. 本节分别测试了利用 *HMAC* 和数字签名实现以上两种证明过程的时间开销, 证明过程中所生成的 *HMAC* 利用 HMAC-SHA256 算法生成, 生成 *HMAC* 的对称密钥长度为 256 位, 而数字签名使用 SM2-256 算法和 RSA-2048 算法生成.

基于数字签名的物联网设备平台状态证明过程包括主机生成并发送随机证明挑战给物联网设备、物联网设备基于收到的随机挑战生成签名并将其和作为自身平台状态证据的复合设备标识符 *CDI* 返回给主机以及主机验证物联网设备的签名和平台状态证据 *CDI*. 而基于 *HMAC* 的物联网设备平台状态证明过程包括主机生成并发送远程证明挑战给物联网设备、物联网设备基于收到的随机证明挑战生成 *HMAC* 并将其和复合设备标识符 *CDI* 返回给主机以及主机验证物联网设备生成的 *HMAC* 和平台状态证据 *CDI*.

本节分别对基于 *HMAC* 和基于数字签名的物联网设备平台状态证明过程进行了 10 次测试, 结果如图 13 所示. 基于 HMAC-SHA256 算法的平均证明时间开销为 121.734 ms, 同时基于 SM2-256 算法的平均证明时间开销为 321.011 ms, 而基于 RSA-2048 算法的平均证明时间开销为 525.088 ms. 由实验结果分析可知, 基于数字签名的物联网设备平台状态证明过程的平均时间开销高于基于 HMAC-SHA256 算法的平均证明时间开销, 而基于 RSA-2048 算法的平均证明时间开销高于基于 SM2-256 算法的平均证明时间开销. 除物联网设备与主机之间的数据传输外, 基于数字签名的物联网设备平台状态证明的时间开销主要来自于生成签名和验证签名操作.

基于数字签名的物联网设备与主机之间的双向身

份认证过程包括物联网设备生成随机证明挑战并发送给主机、主机基于收到的随机证明挑战生成签名并将其和自身生成的随机证明挑战发送给物联网设备、物联网设备对主机生成的签名进行验证并将基于主机发送的随机证明挑战生成的签名返回给主机以及主机验证物联网设备生成的签名. 基于 *HMAC* 的双向身份认证过程包括物联网设备生成并发送随机证明挑战给主机、主机基于收到的随机证明挑战生成 *HMAC* 并将其与自身生成的随机证明挑战返回给物联网设备、物联网设备验证主机发送的 *HMAC* 并将基于主机发送的随机证明挑战所生成的 *HMAC* 返回给主机以及主机验证物联网设备生成的 *HMAC*.

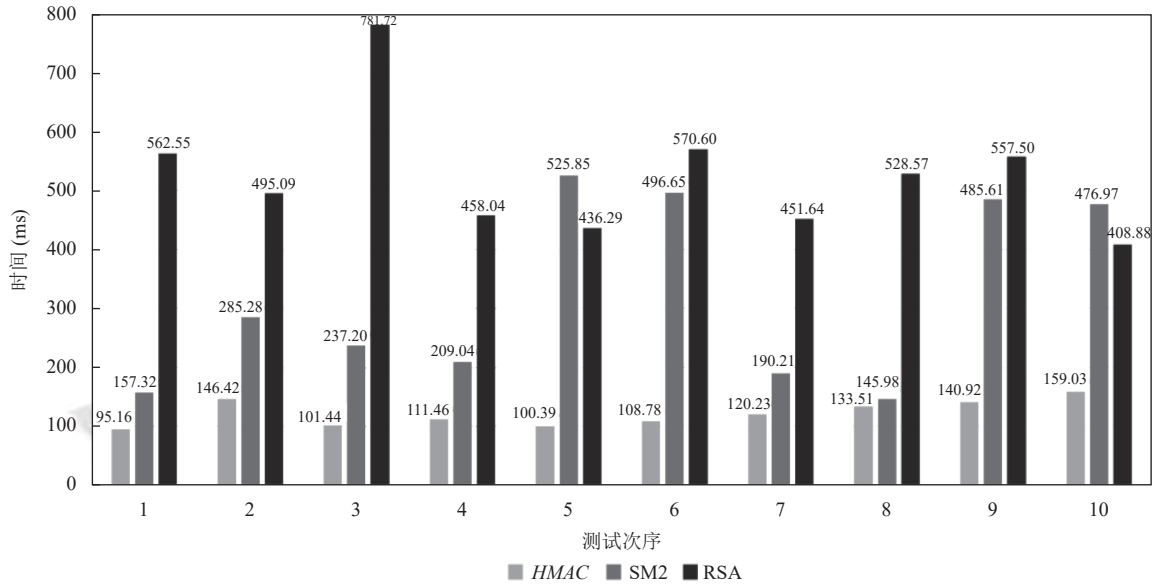

图 13 物联网设备平台状态证明时间

本节分别测试了 10 次利用 *HMAC* 和数字签名实现物联网设备与主机之间双向身份认证的时间开销, 测试结果如图 14 所示. 基于 HMAC-SHA256 算法的双向身份认证的平均时间开销为 215.288 ms, 基于 SM2-256 算法的双向身份认证的平均时间开销为 437.258 ms, 而基于 RSA-2048 算法的双向身份认证的平均时间开销为 680.223 ms. 由实验结果分析可知, 基于数字签名的双向身份认证的平均时间开销高于基于 HMAC-SHA256 的双向身份认证, 基于 RSA-2048 算法的双向身份认证的平均时间开销高于基于 SM2-256 算法的双向身份认证. 除数据传输以外, 基于数字签名的双向身份认证的主要时间开销同样来自于证明过程中生成签名、验证签名操作.

(3) 安全存储的性能开销

本节测试了安全存储模块的文件上传与文件下载的性能, 分别测试了 10 次主机上传 16 KB (16 384 字节) 文件至物联网设备加密分区中和主机从物联网设备加密分区中下载 16 KB 文件的时间开销, 测试结果如图 15、图 16 所示. 平均文件上传和平均文件下载时间分别为 290.719 ms 和 145.938 ms, 平均文件上传速率和下载速率分别约为 0.451 Mb/s 和 0.898 Mb/s. 文件上传的时间开销主要来自于数据传输和加密分区写操作, 文件下载的时间开销主要来自于数据传输和加密分区读操作.

5.4 安全性分析

当攻击者篡改了物联网设备的固件时, 物联网设备在可信启动阶段无法生成正确的复合设备标识符 *CDI* 和远程证明密钥, 因此主机在验证物联网设备的平台身份和平台状态时会判断出此时物联网设备的平台身份或平台状态至少有一个是不可信的.

当攻击者试图冒充可信物联网设备或合法主机的身份时, 由于攻击者无法获得物联网设备和主机的远程证明密钥, 因此无法生成正确的远程证明报告. 而物联网设备和主机会通过合法的远程证明密钥检测出错误的远程证明报告, 从而判断出攻击者的非法身份.

当主机上的非法用户试图访问加密分区时, 由于非法主机用户无法获得正确的加密分区访问密码, 而物联网设备会通过验证访问密码识别出非法的主机用户.

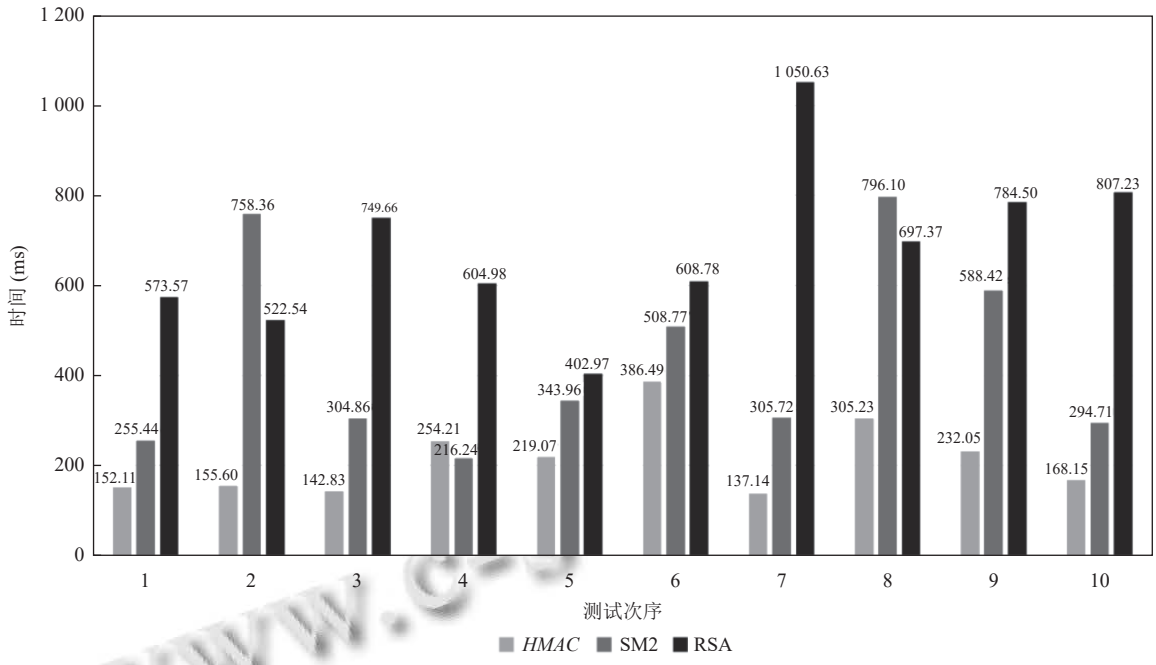

图 14 物联网设备与主机双向身份认证时间

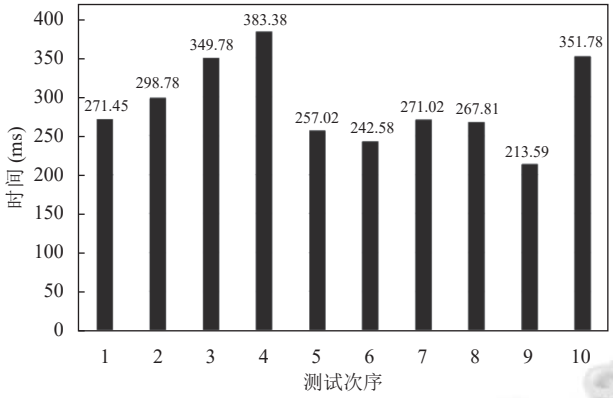

图 15 文件上传时间

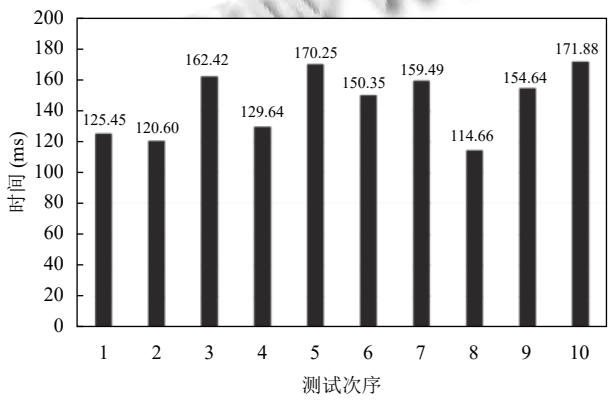

图 16 文件下载时间

当外置移动存储设备处于丢失或失窃等非安全状态时, 由于只有拥有合法密钥加密密钥的可信物联网设备才能打开该外置移动存储设备中的加密分区, 因此存储在其中的加密数据不会泄漏.

6 总结与展望

本文提出了一种基于轻量级信任根 DICE 的证明存储方案, 利用基于轻量级信任根 DICE 构建的可信物联网设备提供安全存储服务, 通过将存储数据的加解密操作移至可信物联网设备上执行的方式消除因主机遭受各类攻击对存储数据造成的威胁. 同时构建基于信任根 DICE 的物联网设备远程证明机制和访问控制机制实现安全认证和安全交互通道的建立, 从整体上提高方案的安全性. 并且提出了一种基于 DICE 的树形度量引导模型, 可减少不必要的固件度量引导开销. 实验结果表明该方案具有较好的可信启动、远程证明和安全存储性能, 在实际应用过程中可以满足通用场景下的远程证明和安全存储需求, 具备一定的应用价值. 在未来, 主要的工作是将基于 DICE 的物联网设备可信启动扩展到操作系统内核级, 进一步提高物联网设备在引导启动过程中的安全性.

参考文献

1 Yang P, Xiong NX, Ren JL. Data security and privacy

protection for cloud storage: A survey. IEEE Access, 2020, 8: 131723–131740. [doi: 10.1109/ACCESS.2020.3009876]
2 Meijer C, van Gastel B. Self-encrypting deception: Weaknesses in the encryption of solid state drives. Proceedings of the 2019 IEEE Symposium on Security and Privacy. San Francisco: IEEE, 2019. 72–87.
3 Tan C, Zhang LJ, Bao L. A deep exploration of BitLocker encryption and security analysis. Proceedings of the 20th IEEE International Conference on Communication Technology. Nanning: IEEE, 2020. 1070–1074.
4 Wu M, Lu TJ, Ling FY, *et al*. Research on the architecture of Internet of Things. Proceedings of the 3rd International Conference on Advanced Computer Theory and Engineering. Chengdu: IEEE, 2010. V5-484–V5-487.
5 Khan MA, Salah K. IoT security: Review, blockchain solutions, and open challenges. Future Generation Computer Systems, 2018, 82: 395–411. [doi: 10.1016/j.future.2017.11.022]
6 Siddiqui AS, Gui YT, Saqib F. Secure boot for reconfigurable architectures. Cryptography, 2020, 4(4): 26. [doi: 10.3390/cryptography4040026]
7 Thakor VA, Razzaque MA, Khandaker MR. Lightweight cryptography algorithms for resource-constrained IoT devices: A review, comparison and research opportunities. IEEE Access, 2021, 9: 28177–28193. [doi: 10.1109/ACCESS.2021.3052867]
8 Khalid O, Rolfes C, Ibing A. On implementing trusted boot for embedded systems. Proceedings of the 2013 IEEE International Symposium on Hardware-oriented Security and Trust. Austin: IEEE, 2013. 75–80.
9 Helble SC, Kretz ID, Loscocco PA, *et al*. Flexible mechanisms for remote attestation. ACM Transactions on Privacy and Security, 2021, 24(4): 29.
10 Jäger L, Petri R, Fuchs A. Rolling DICE: Lightweight remote attestation for COTS IoT hardware. Proceedings of the 12th International Conference on Availability, Reliability and Security. Reggio: ACM, 2017. 95.
11 冯登国, 刘敬彬, 秦宇, 等. 创新发展中的可信计算理论与技术. 中国科学: 信息科学, 2020, 50(8): 1127–1147.
12 Mayes KE, Markantonakis K. Smart Cards, Tokens, Security and Applications. New York: Springer, 2010.
13 Tao Z, Rastogi A, Gupta N, *et al*. DICE*: A formally verified implementation of DICE measured boot. Proceedings of the 30th USENIX Security Symposium. USENIX Association, 2021. 1091–1107.
14 Seshadri A, Perrig A, van Doorn L, *et al*. SWATT: Software-based attestation for embedded devices. Proceedings of the 2004 IEEE Symposium on Security and Privacy. Berkeley: IEEE, 2004. 272–282.
15 Spinellis D. Reflection as a mechanism for software integrity verification. ACM Transactions on Information and System Security, 2000, 3(1): 51–62. [doi: 10.1145/353323.353383]
16 Costan V, Devadas S. Intel SGX explained. IACR Cryptology ePrint Archive, 2016: 86.
17 Brasser F, El Mahjoub B, Sadeghi AR, *et al*. TyTAN: Tiny trust anchor for tiny devices. Proceedings of the 52nd ACM/EDAC/IEEE Design Automation Conference. San Francisco: IEEE, 2015. 1–6.
18 Koeberl P, Schulz S, Sadeghi AR, *et al*. TrustLite: A security architecture for tiny embedded devices. Proceedings of the 9th European Conference on Computer Systems. Amsterdam: ACM, 2014. 10.
19 Yang Y, Wang XR, Zhu SC, *et al*. Distributed software-based attestation for node compromise detection in sensor networks. Proceedings of the 26th IEEE International Symposium on Reliable Distributed Systems. Beijing: IEEE, 2007. 219–230.
20 Park T, Shin KG. Soft tamper-proofing via program integrity verification in wireless sensor networks. IEEE Transactions on Mobile Computing, 2005, 4(3): 297–309. [doi: 10.1109/TMC.2005.44]
21 Sabt M, Achemlal M, Bouabdallah A. Trusted execution environment: What it is, and what it is not. Proceedings of the 2015 IEEE Trustcom/BigDataSE/ISPA. Helsinki: IEEE, 2015. 57–64.
22 Yang K, Chen LQ, Zhang ZF, *et al*. Direct anonymous attestation with optimal TPM signing efficiency. IEEE Transactions on Information Forensics and Security, 2021, 16: 2260–2275. [doi: 10.1109/TIFS.2021.3051801]
23 Anati I, Gueron S, Johnson S, *et al*. Innovative technology for CPU based attestation and sealing. Proceedings of the 2nd International Workshop on Hardware and Architectural Support for Security and Privacy. 2013.
24 TCG. Symmetric identity based device attestation. https://trustedcomputinggroup.org/wp-content/uploads/TCG_DICE_SymIDAttest_v1_r0p94_pubrev.pdf. (2019-07-24).
25 Ahn J, Lee J, Ko Y, *et al*. DISKSHIELD: A data tamper-resistant storage for Intel SGX. Proceedings of the 15th ACM Asia Conference on Computer and Communications Security. Taipei: ACM, 2020. 799–812.

(校对责编: 孙君艳)