计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2022,31(9):183−191 [doi: 10.15888/j.cnki.csa.008671]

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

面向物联网设备的安全集群证明及修复协议①

林江南[1], 吴秋新[1], 冯　伟[2]

[1](北京信息科技大学 理学院, 北京 100192)
[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: 林江南, E-mail: 15611007977@163.com

摘　要: 由于物联网设备本身缺少安全机制, 物联网环境面临着严峻的安全挑战. 而远程证明能够认证设备真实性和完整性, 可以通过远程方式建立对物联网设备的信任. 集群证明是远程证明技术的扩展, 可以适用于大量设备构成的集群. 相较于传统的远程证明, 集群证明解放了验证设备, 提高了验证的效率. 目前, 集群证明方法主要是针对静态网络, 而且对于受损设备也缺乏高效的修复机制. 针对这些问题, 本文提出了一种基于信誉机制和 Merkle 树的安全集群证明及修复方法. 首先, 本文方法使用信誉机制实现了多对一的证明协议, 能有效解决单点故障, 从设备触发验证, 并且能够适用于半动态网络. 其次, 本文引入 Merkle 树进行度量, 能够快速地识别被感染的代码块, 并进行高效地恢复; 最后, 本文对提出的集群证明方法进行了安全性分析和性能评估, 结果表明, 本文集群证明在提高了安全性的同时导致的性能开销是可以接受的.

关键词: 物联网安全; 可信计算; 远程证明; 集群证明; 信誉机制; 默克尔树

引用格式: 林江南,吴秋新,冯伟.面向物联网设备的安全集群证明及修复协议.计算机系统应用,2022,31(9):183–191. http://www.c-s-a.org.cn/1003-3254/8671.html

Secure Swarm Attestation and Recovery Scheme for IoT Devices

LIN Jiang-Nan[1], WU Qiu-Xin[1], FENG Wei[2]

[1](School of Applied Science, Beijing Information Science and Technology University, Beijing 100192, China)
[2](Trusted Computing and Information Assurance Laboratory, Institute of Software, Chinese Academy of Sciences, Beijing 100190, China)

Abstract: Owing to the lack of security mechanisms for Internet of Things (IoT) devices, the IoT environment faces serious security challenges. However, remote attestation can identify the authenticity and integrity of devices and can also establish trust in IoT devices through a remote mode. Swarm attestation is an extension of remote attestation technology, which can be applied to swarm composed of a large number of devices. Compared with the traditional remote attestation, the swarm attestation liberates the verifier and improves verification efficiency. At present, the swarm attestation is mainly used for static networks, and there is no efficient recovery mechanism for compromised devices. To solve these problems, this study proposes a secure swarm attestation and recovery scheme based on reputation mechanism and Merkle tree. Firstly, we use the reputation mechanism to achieve a many-to-one attestation scheme, which can effectively solve the single point of failure and also trigger the attestation from the device. In addition, the attestation scheme is suitable for semi-dynamic networks. Secondly, we introduce the Merkle tree for measurement, which can quickly and accurately identify the code blocks compromised by malicious software and efficiently recover them. Finally, the security analysis and performance evaluation of the swarm attestation scheme are presented. The results show that the swarm attestation in this study improves the security, and its performance overhead is acceptable.

Key words: Internet of Things (IoT) security; trusted computing; remote attestation; swarm attestation; reputation management; Merkle tree

① 基金项目: 国家重点研发计划 (2020YFE0200600)
收稿时间: 2021-12-01; 修改时间: 2021-12-29; 采用时间: 2022-01-20; csa 在线出版时间: 2022-06-28

1 引言

在物联网发展迅速的今天, 物联网设备早已遍布人们日常生活的周围. 物联网设备通常使用低功耗的嵌入式智能设备, 用于执行一些特定的任务, 比如发送、传输以及处理一些在环境中所获得的数据[1], 从而构成了物联网系统, 以此提高整个网络的运行速度和服务质量.

由于嵌入式物联网设备的低成本, 大量的物联网设备应用于日常生活中, 但其内存、计算能力等方面都存在限制, 特别是缺乏安全机制[2]. 从而导致嵌入式设备系统容易成为网络攻击的目标, 如 Mirai[3], 将僵尸程序传播至网络, 感染在线设备, 从而形成僵尸网络, 导致很多网络服务功能瘫痪, 无法进行正常工作, 从而造成无法挽回的损失.

而远程证明能够证明设备完整性这一特性, 恰好可以用于解决嵌入物联网设备的安全问题. 利用远程证明基于证明者提供的可信度建立起一个安全的交互式协议[4]. 目前学术界已经提出了很多的远程证明协议, 大体可以分为 3 类: (1) 基于软件的证明, 无需硬件支持, 但往往基于实践中难以实现的假设, 如 SAKE[5]、VIPER[6]; (2) 基于硬件的证明, 由于其昂贵且复杂的特性, 对于资源受限的嵌入式物联网设备不太实用, 如 TrustVisor[7]、Flicker[8]; (3) 混合证明, 旨在提供最小化的硬件支持来弥补纯软件证明的安全性不足, 实现与基于硬件相似的安全保护, 如 TrustLite[9]、TyTAN[10].

本文提出了一种基于信誉机制和 Merkle 树的安全集群证明及修复方案, 主要贡献如下:

(1) 本文方法使用信誉机制实现了多对一的证明协议, 能有效解决单点故障, 消除了固定的验证设备, 可以从设备触发验证, 使得证明更加及时, 并且适用于半动态网络.

(2) 引入 Merkle 树进行度量, 能够快速精确地判断出被恶意软件感染的代码块, 再形成定制的补丁进行恢复, 不仅减少数据传输, 还能高效地修复受损设备.

(3) 本文还对提出的集群证明方法进行了安全性分析和性能评估, 结果表明, 本文集群证明在提高了安全性的同时所导致的性能开销是可以接受的.

本文结构如下: 第 2 节介绍集群证明的相关工作; 第 3 节给出本文基于的系统模型与假设; 第 4 节详细给出本文提出的集群证明协议与修复机制; 第 5 节对本文提出的协议进行安全性分析; 第 6 节为性能评估, 包括计算、通信、内存、运行时间以及能源开销方面的分析、本方案与 ESDRA[11] 及 HEALED[12] 的安全性对比以及仿真模拟的实验结果; 第 7 节对本文工作进行总结.

2 相关工作

集群证明. 纵观集群证明的发展历程, 随着第一个集群证明协议 SEDA[13] 提出至今, 不少集群证明的方案呈现在公众的视野中. 大多数都是以 SEDA 为基础的一对多证明协议. 这些证明协议以验证者为根构造生成树, 从根往下进行证明, 最后不断将证明结果汇聚至验证者, 这也就意味着这类协议只能应用于静态网络. 比如, SeED[14] 在 SEDA 的基础上增加了抵抗 DoS 攻击, 从而形成了非交互式的拒绝服务攻击的认证方案. 在 2019 年, ESDRA 提出了第一个多对一的集群证明协议, 通过设备端发起的证明, 从而可以适用于半动态的网络. POSTER[15], SALAD[16] 都是基于设备的自身认证, 积累个体验证报告, 最后共享至整个网络, 从而实现适用于动态网络的认证协议.

安全修复. 目前大多数协议只提出了证明方案, 对于后续的修复没有详细的介绍.已有的修复协议方案, 如 HEALED, 利用集群相同软件配置的可信设备对其进行修复, 从而将受损设备回滚至可信状态.

3 系统模型与假设

在大规模的集群S中包含着软硬件配置不同的异构设备, 并且在通信过程中, 能耗随距离增加而急剧增加. 面对着无处不在的各式攻击, 如果让管理者一一验证, 不仅存在验证不及时的风险, 并且验证者的性能将成为整体认证方案效率的瓶颈. 为了提高验证效率以及降低功耗, 我们将集群S中的每个设备按照通信距离分成若干个簇, 每个簇都有一个簇头节点. 在簇内, 各个设备节点之间相互证明从而构建出集群的可信环境. 此外, 检测到受损设备应当给予修复的机会, 若是修复不了, 即可判定修复成本大于他自身的价值, 可以进行移除操作. 当然, 为了识别集群中由于遭受物理攻击而探测不到的设备, 实施在线探测及时识别可能受到攻击的设备, 并进行隔离验证.

在本文集群证明系统中, 主要的参与者包含: 网络管理者M, 负责初始化以及修复集群中受损的设备节点; 簇头节点D_i, 负责管理簇内设备; 组内普通节点D_{ij},

由簇头节点D_i进行管理.

本文方法基于以下几点假设:

(1) 假设所有节点都满足安全远程证明的最低硬件要求, 即只读存储器和简单的内存保护单元, 可以通过 SMART[17] 等机制实现.

(2) 假设证明例程为原子程序运行, 并且证明代码无法被修改.

(3) 为了确保证明结果的可靠性, 假设每个节点应至少有 3 个邻居设备.

(4) 假设物理攻击会导致设备一段时间不可用, 从而攻击期间无法探测到设备.

(5) 由于 DDoS 攻击几乎不可能被完全抵抗, 与其他集群证明方案一样, 本文不考虑 DDoS 攻击.

4 系统方案

本方案提出一个多对一的集群证明及修复方案. 如图 1 所示, 方案可分为两个阶段: 离线阶段和在线阶段. 在离线阶段完成设备的初始化以及设备之间的连接过程, 从而构建出整个集群的网络. 而在线阶段主要完成对设备的验证以及发现受损设备后进行的修复协议. 此外, 在线阶段还加入了缺失探测机制, 可以及时发现因物理攻击导致不可达的节点, 将其隔离出集群, 再进行修复或者彻底移除操作. 表 1 定义了本文所用的符号及参数.

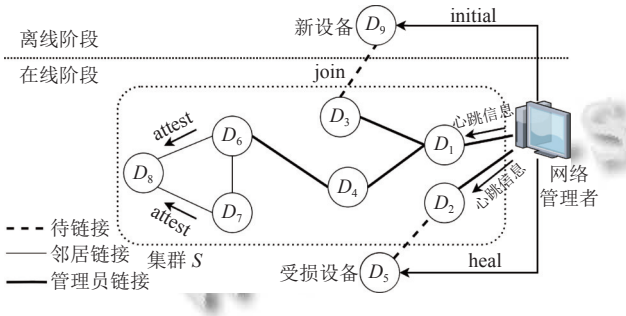

图 1 协议网络图

4.1 离线阶段

离线阶段包括设备的初始化 (initial 协议) 以及设备连接 (join 协议). 设备初始化过程中, 网络管理者对新增设备进行初始化工作; 设备连接过程是设备之间通过网络管理者提供的安全通道建立连接, 为后续证明及信息交流提供基础.

4.1.1 设备初始化

如图 2 所示, 当新设备D_7要加入网络时, 网络管理者M就会对D_7执行initial协议, 生成设备对应的配置信息, 其中包括: 私钥sk_7, 公钥pk_7, 身份证书$cert(pk_7)$, 代码证书$cert(c_7)$, 网路管理员公钥pk_M, 初始信誉值w_7, 最后验证时间t_{a_7}以及唯一设备标识符d_7, M保存软件配置代码c_7, 便于后续修复 (heal) 协议的使用. 对新设备的初始化可以表述为:

$$initial(D_i:c_i,1^{\ell};M:pk_M)\rightarrow (D_i:sk_i,pk_i,cert(pk_i),cert(c_i),pk_M,w_i,t_{a_i},d_i;M,c_i)$$

表 1 符号和参数

类别	变量名	描述信息
实体	M	网络管理者
	D_i	第i个簇头节点
	D_{ij}	第i簇的第j个普通节点
集群参数	s	集群S中的设备数量
	n	集群S中簇的数量
	n_i	第i簇中设备数量
	m_i	证明者的相邻节点数量
设备参数	sk	私钥
	pk	公钥
	c	平台软件配置
	$cert(pk)$	身份证书
	$cert(c)$	代码证书
	t_a	最后验证时间
	t_e	全局验证有效时间
	k	对称密钥
	w	信誉值
	d	唯一设备标识符
协议参数	b	临时证明结果
	N	随机数
	λ	调和参数
	f	最终验证结果
	Γ	修复补丁
	r	修复结果
	$w_{\min}$	信誉值阈值
	$w_{\max}$	最大信誉值
	w_{rwd}	奖励信誉值
	w_{psm}	惩罚信誉值
程序	$mac()/Vermac()$	生成/验证一个MHTMAC
	$Sign()/VerSign()$	对结果进行签名/验证签名
	$encrypt()/decrypt()$	加密/解密操作
	$DoPatch()$	安装补丁
	$attest()$	运行证明例程

每个设备的公私钥对都是基于 SM2 ECC 密钥体制[18] 生成. 代码证书$cert(c_i)$包含设备的可信散列代码c_i以及证书的公共信息. t_{a_i}和t_e主要是在证明阶段提供可靠的标准, 当前时刻为$t_{a_i}+t_e$时, 就会对该设备执行设备证明 (attest 协议). 由于触发验证条件在于设备本身, 集群内就可以并发执行attest协议. 此外, 每个设备

的信誉值取决于它参与的每次证明例程. 在证明例程中, 任何一个节点没有权限为其他节点报告信誉值. 一旦发现某个节点是不可信的, 其信用值将被设置为$-w_{\max}$. 在我们的方案中, 信誉值更高节点在邻居节点的最终证明结果生成中拥有更大的权重.

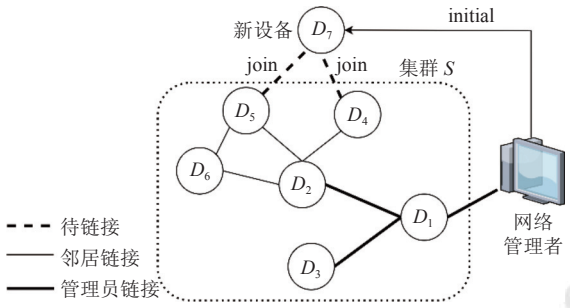

图 2 离线阶段

4.1.2 设备连接

当设备D_i初始化或者更新后, 通过网络管理者M提供的安全通道, 与邻居设备进行配置信息的交换. 如图 2 所示, D_7初始化后, 与邻居设备D_4和D_5进行连接. 在连接过程中, 它们双方交换自身的公钥pk_i、身份证书$cert(pk_i)$、代码证书$cert(c_i)$、当前信誉值w_i、最后验证时间t_{a_i}以及唯一设备标识符d_i. 设备的配置信息存储在受硬件保护的内存之中, 这就意味着设备节点无法虚报配置信息. 通过已认证的密钥协商协议生成基于双方私钥的会话密钥k, 在本文中使用非相邻形式椭圆曲线 Diffie-Hellman 密钥交换协议[19]. 在后面的证明和修复过程中, k是双方进行数据交流的保证. 设备连接过程可描述为:

$$join[D_i:sk_i,D_j:sk_j,*:cert(pk_i),cert(pk_j),cert(c_i),cert(c_j),d_i,d_j,w_i,w_j]\rightarrow[D_i:k_{ij},D_j:k_{ij}]$$

网络管理者M会周期性地根据簇头选择算法来选取簇头, 使得簇头的分布能覆盖整个集群, 本文采用基于最佳簇半径的无线传感器网络分簇路由算法[20]. 簇头节点通过信息的传输形成了以M为根节点的树形结构, 其他普通设备节点以传输距离为参照连接到对应的簇, 从而形成集群. 普通设备保存簇头的公钥pk_M, 便于后期提交证明报告. 对于网络管理者M, 存储集群中所有设备当前的信誉值, 在每次证明例程后会更新对应设备的信誉值. 当簇头发现不可信的设备节点时, 其信誉值将被设为$-w_{\max}$, 将w_i以及d_i发送给M, 然后M对其进行隔离, 再执行heal协议或者移除网络等操作.

4.2 在线阶段

在线阶段包括对设备进行验证 (attest)、对设备的修复 (heal) 以及缺失探测 3 个子协议. 一旦有设备D_i触发验证条件, 邻居设备就会对其执行attest协议, 每个邻居生成临时的证明结果上传至簇头, 由簇头生成最终的验证结果, 最后将D_i的最新信誉值上报网络管理者M. M接收到D_i的最新信誉值, 若等于信誉值最大值$-w_{\max}$, M就会对D_i执行heal协议, D_i将最终修复结果反馈M. 在缺失探测中采用心跳协议来进行, M广播心跳请求, 由簇头节点负责收集反馈, 找出不可达节点, 从而对其进行修复或移除等操作.

在此, 我们引入 Merkle 树对设备当前的软件状态进行度量. 由于 Merkle 树能够自下而上计算度量值, 从而汇聚到根节点. 因此, 在证明阶段, 邻居设备仅需验证根节点的度量值即可形成临时验证报告. 在修复阶段, 如图 3 所示, 证明设备D_i将待证明的代码分成ω段等长的部分: $c_1,c_2,\cdots,c_\omega$, 并对应计算哈希值: $h_i[\omega+1]$, $h_i[\omega+2],\cdots,h_i[2\omega]$, 然后以这些哈希值为叶子节点, c_i'为根, 构造 Merkle 哈希树, 其中, 2ω表示 Merkle hash tree (MHT) 中除根节点之外的节点数. 受恶意软件感染的代码段会导致沿着根路径生成错误的哈希值. 通过这种方式, 可以确认受恶意软件感染的代码段.

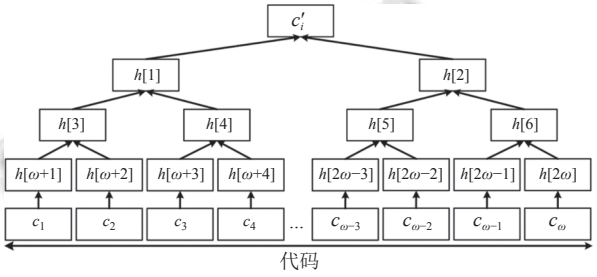

图 3 Merkle 哈希树

4.2.1 设备证明

每当设备D_i的信誉值$w_i<w_{\min}$或者到达验证时刻, 邻居设备D_j就会对D_i发出挑战, 并记录发送时间, 挑战包含随机数N_{ij}. D_i结合挑战和自身当前状态c_i'以及会话密钥k_{ij}生成消息验证码μ_{ij}来报告其软件状态, 在此采取基于 MHT 进行度量, 最终度量为以c_i'为根节点的树. D_j接收到结果后, 通过连接阶段保存的c_{ij}以及密钥k_{ij}验证μ_{ij}的正确性, 若μ_{ij}验证成功, 则D_j生成临时验证结果$b_{ij}=1$, 反之则推断D_i受到攻击, 即$b_{ij}=-1$, 而当D_i无响应, 即为超时, 则$b_{ij}=0$. 形式化为:

$$attest[D_i:c_i',cert(c_i),k_{ij};D_j:N_{ij},c_{ij},k_{ij},*:-]\rightarrow [D_i:\mu_{ij},D_j:b].$$

簇头接收所有邻居节点的临时证明结果后, 通过信誉机制进行最终验证结果, 每个邻居设备的信誉值代表着其参与生成最终验证结果的权重. 最后广播D_i的唯一设备标识d_i以及当前的信誉值w_i. 对于邻居设备D_j, 簇头会将最终验证结果f与各邻居节点的临时证明作对比, 做出相对应的奖惩措施. 最终验证结果及新信誉值计算公式如下:

$$f=\begin{cases}1, & \dfrac{\sum_{j=1}^{x}(b_{ij}w_j)}{\lambda w_{\text{med}}}\geqslant 1\\ -1, & \dfrac{\sum_{j=1}^{x}(b_{ij}w_j)}{\lambda w_{\text{med}}}<1\end{cases}$$

$$w_i=\begin{cases}(w_{\max}-w_{\min})\dfrac{\sum_{j=1}^{x}(b_{ij}w_j)}{w_{\text{med}}}+w_{\min}, & f=1\\ -w_{\max}, & f=-1\end{cases}$$

其中, x为D_i邻居数量, w_{med}为邻居节点信誉值中位数, 它能够对于高信誉值的设备起到制衡作用, λ为调和参数, 且$\lambda\in(0,1]$, 随λ增加验证的条件越是严格. attest协议流程如算法 1 所示.

算法 1. 邻居设备D_j对D_i的attest算法

输入: N_{ij}; c_i'; c_{ij}; k_{ij}
输出: b

```
//D_j运行程序
发送req_ij=mac(k_ij;N_ij), N_ij给D_i;
startTimer();
if Timerout()=1 then b_ij=0;
else if Vermac(k_ij;N_ij||c_ij,μ_ij)=1 then b_ij=1;
else b_ij=−1;
end if
将b_ij发送至簇头
//D_i运行程序
if Vermac(k_ij;|req_ij)=1 then
    μ_ij=mac(k_ij;N_ij||c_i')
end if
```

4.2.2 设备修复

网络管理者M接收簇头上报证明结果, 包括证明设备D_i的更新信誉值w_i以及唯一设备标识符d_i. 当$w_i=-w_{\max}$, M对D_i执行 heal 协议. 具体流程如下.

由于每个设备都有M的公钥, 因此M可以在集群中与任意设备进行通信, M向D_i发送一个请求信息req_i. D_i接收后将其软件配置c_i'和一个新的随机数N_i利用 MHT 算法度量后用M的公钥加密后发送给M. M将自己保存的软件配置c_i以同样方式度量进行 MHT 的遍历, 直到到达叶子节点. 最后, M为每个异常的叶子节点添加一个代码段l, 用自身私钥sk_M对l进行签名生成补丁Γ, 并将其发送回D_i. 一个代码段$l=\{s,e,p\}$包含其起始地址s、结束地址e, 其代码p. D_i进行补丁修复, 修复成功输出$r=1$. 否则, 输出$r=0$. heal过程可以通过如下公式表述为:

$$heal[M:sk_M,c_i,\{h_M[0],\cdots,h_M[2\omega]\};D_i:pk_M,c_i',\{h_i[0],\cdots,h_i[2\omega]\};*:-]\rightarrow[M,N_i,\Gamma;D_i:r]$$

D_i可以谎报修复结果来躲避补丁修复, 因此, 当heal结束后, 将D_i的信誉值设置为 0, 这样D_i就会触发attest协议, 对D_i进行再次证明, 若证明可信, 则说明D_i已经恢复正常的软件配置. 反之, D_i的信誉值将被设置为$-w_{\max}$, 从而需要进行再次修复, 直到其配置正常. 此外, 为了避免同一设备attest与heal反复执行, M记录设备执行heal的时间, 如果出现连续 3 次执行heal失败, 将直接移除该设备. heal协议的算法流程如算法 2 所示.

算法 2. 网络管理者M对D_i的heal算法

输入: N_i; c_i'; $\{h_i[0],\cdots,h_i[2\omega]\}$; pk_M; c_i; $\{h_M[0],\cdots,h_M[2\omega]\}$
输出: r

```
1:  //M运行程序
2:  发送req_i=Sign(sk_M;N_i)给D_i;
3:  wait() //等待接收反馈
4:  if decrypt(sk_M;d_i')=1 then
5:      while (k<2ω) then
6:          if (h_i[k]!=h_M[k]) then
7:              k=2k+1;
8:          else
9:              k++;
10:         end if
11: end if
12: l={s,e,p};
13: 将Γ=Sign(sk_M;l)发送至D_i
14: wait() //等待D_i回复后
15: if r=0 then
16:     attest();
17: end if
18: //D_i运行程序
19: if VerSign(pk_M;N_i||req_i)=1 then
20:     d_i=MHT(c_i');
21:     将d_i'=encrypt(pk_M;d_i)发送至M
22: end if
23: wait() //等待接收补丁
24: if VerSign(pk_M;Γ)=1 then
25:     DoPatch(Γ);
```

```
26:        r=1;
27: else
28:        r=0;
29: end if
30: 将r加密后发往M
```

4.2.3 在线探测

在线探测阶段负责对整个集群设备进行定期的探测, 监测设备的运行状态. 本文采用基于多级心跳协议[21], 由网络管理者定期给簇头节点发出心跳信息, 簇头节点在组内广播心跳信息, 并收集组内普通节点的心跳信息, 形成一个 alive-node 消息, 发给网络管理者. 若在规定的时间未接收到节点设备的心跳信息, 利用多级心跳协议增加和删除节点的便利性, 将其从集群的拓扑网络中删除, 经过验证为可信后, 再重新连接进入集群中.

5 安全性分析

我们将这个过程形式化为一个安全实验EXP_{ADV}敌手ADV可以与其相连的设备进行通信, 至少修改一个设备D_i的软件配置, ADV可以篡改、窃听或删除所有通过D_i传输的信息. 经过一个多项式步骤后, D_i被验证簇头最终验证为 1, 即为证明成功. 也就是敌手成功攻破D_i.

定义 1. 安全集群证明. F是一个关于$\ell_N,\ell_{\text{sign}},\ell_{\text{hmac}}$的多项式函数. 受损设备通过证明的概率为$Pr[f=1|EXP_{ADV}(1^{\ell})=b]$在$\ell=F(\ell_N,\ell_{\text{sign}},\ell_{\text{hmac}})$中被认为是可以忽略的, 则证明和修复方案是安全的.

定理 1. 如果底层签名、加密和 MHTMAC 方案是选择性防伪的, 则本方案是一个安全的集群证明方案.

证明: ADV可以通过欺骗邻居节点返回$b=1$或欺骗网络管理者返回$r=1$来破坏本方案的安全. 下面通过两种情况来分析:

(1) ADV攻击$attest$: ADV想要使得$b=1$来通过邻居节点的验证可以使用以下策略: 1) 使用基于原始代码的先前的 HMAC μ_{old}; 2) 计算出基于新挑战的 HMAC μ'; 3) 入侵多个邻居节点, 影响最终的证明结果.

对于策略 1), 每次的挑战N都是新的随机值, 当且仅当$N=N_{\text{old}}$时, 验证可以通过. 然而$N=N_{\text{old}}$的概率为$2^{-\ell_N}$. 所以通过验证的概率可以忽略. 对于策略 2), 一旦软件配置发生改变生成的 HMAC 无法通过邻居节点的验证, 因此μ'通过验证的概率也可以忽略. 对于策略 3), 对于单个节点而言, $Pr[f=1|EXP_{ADV}(1^{\ell})=b]$是可以忽略的, 而同时攻破$k$个节点概率为$(Pr[f=1|EXP_{ADV}(1^{\ell})=b])^k$, 也是可以忽略的.

(2) ADV攻击heal: heal协议由网络管理者M对受损设备D_i进行修复, ADV可以以下策略来逃避heal协议: 1) 伪造$r=1$返回给M; 2) 在D_i安装补丁前篡改补丁.

对于策略 1), 虽然可以逃避补丁Γ安装, 但是在之后的证明例程依旧被检测出来, 超过 3 次将被移出集群. 对于策略 2), 与attest类似, ADV试图提取Γ中提取补丁代码再进行修改的概率是可以忽略的.

因此, ADV通过攻击attest协议heal协议来攻破良性设备的概率在$\ell_N,\ell_{\text{sign}},\ell_{\text{hmac}}$上可以忽略不计, 再加上在线探测阶段对物理攻击进行检测, 由此可证明本方案是一个安全的集群证明方案.

6 性能评估

本节将从计算开销、通信开销、内存开销、运行时间以及能源开销这几个方面对本协议进行分析, 此外我们还对本文集群证明协议进行了仿真, 并与其他几个类似方法进行了对比.

(1) 计算开销. 对于普通设备而言, 主要计算开销在于一些密码操作. 在证明阶段, 证明设备需要计算m_i个$mac()$并验证m_i个$Vermac()$, 其中m_i表示D_i邻居设备的数量. 邻居设备验证一个$Vermac()$并计算一个$Sign()$. 对于簇头设备, 需要验证m_i个$VerSign()$并计算 2 个$Sign()$, 一个发往网络管理者M, 一个广播到组内各个节点. 在修复阶段, 修复设备需计算 2 个$encrypt()$验证 2 个$decrypt()$, 而M需要验证 2 个$VerSign()$并计算 2 个$Sign()$.

(2) 通信开销. 我们使用 SM3 实现 MHTMAC, 密钥协商算法和签名方案都是采用基于 SM2 的算法. 以$len(x)=1$表示x的长度为 1 B. 因此, $len(\text{MHTMAC})=32\omega-16$, 其中$\omega$为验证代码分割的段数, $len(Sign)=64$, $len(N)=16$, $len(\mu)=16$, $len(w)=4$, $len(t)=4$, $len(d)=4$, $len(b)=4$, $len(r)=4$. 证书的大小取决于密钥和签名, 所以$len(cert)=48$. 因此, 在证明阶段, 证明设备而言, 需发送$(32\omega-16)m_i$ B 以及接收$16m_i$ B; 邻居设备而言, 需发送 20 B 以及接收$(32\omega-16)$ B; 簇头节点需发送$(128+8n_i)$ B 接收$4m_i$ B; 网络管理者M接收 64 B. 在修复阶段, 修复设备需接收 36 B 以及发送$(32\omega-12)$ B, M发送 36 B 以及接收$(32\omega-12)$ B.

(3) 内存开销. 每个普通设备需要存储: 1) 自己的密钥对(sk_i, pk_i)和设备标识符d_i; 2) 邻居设备的设备标识符d_j及其信用值w_j、最近证明时间t_{a_j}、代码证书$cert(h_j)$和会话密钥k_{ij}; 3) 簇头的公钥. 簇头设备还需要存储簇内所有节点的设备标识、相应的信用值和公钥. 因此, 普通节点需要$(52+80m_i)$ B 的存储空间, 簇头节点需要$(52+80m_i+24n_i)$ B 的存储空间.

(4) 运行时间. 我们使用t_h、t_p、t_{ca}、t_{tr}、t_s、t_v来表示计算或验证 MHTMAC、生成随机数、一跳访问信道并传输一个字节、签名或验证签名的时间. H是群的生成树的高度. 因此, 总证明时间t_{attest}为:

$$t_{\text{attest}} \leqslant m_i(t_h+t_p+t_{ca})+((32\omega-8)m_i+64H)t_{tr} +(m_i+2)t_s+(m_i+n_i+1)t_v.$$

总修复时间t_{heal}为:

$$t_{\text{heal}} \leqslant t_{ca}+t_h+t_p+(32\omega+24)Ht_{tr}+4t_s+4t_v$$

(5) 能源开销. 我们分别使用E_s、E_r、E_p、E_h、E_{sg}、E_v表示发送 1 B、接收 1 B、生成随机数、计算或验证 MHTMAC 以及签名和验证或签名的能源开销. 在一个证明过程中, 簇头节点的最大能耗为:

$$E \leqslant (128+8(n_i+1)E_s)+4m_iE_r+2E_{sg}+t_vm_iE_v$$

证明设备的最大能耗为:

$$E_i \leqslant (32\omega-16)m_iE_s+16m_iE_r+m_iE_h+E_v$$

邻居设备的最大能耗为:

$$E_j \leqslant 20E_s+(32\omega-16)E_r+E_p+E_{sg}+E_v$$

在一个修复过程中, 修复设备的最大能耗为:

$$E_i \leqslant (32\omega+24)E_s+(32\omega+24)E_r+E_p+E_h+4E_{sg}+4E_v$$

本方案还与 ESDRA、SEDA、SALAD 在运行时间、平均能耗等方面进行比较, 具体数据展示如表 2 所示.

(6) 安全性对比. 我们将本方案与 ESDRA、HEALED 进行安全性对比. 如表 3 所示, 在我们的方案中, 增加了在线探测技术, 可以针对物理攻击造成探测不到的设备进行识别, 从而预防物理攻击. 在 ESDRA 中, 簇头与所有簇内设备公用一个会话密码, 存在会话密钥泄露的风险, 本方案改用簇头的公钥进行加密, 仅有簇头本身能够解密, 解决了会话密钥窃取的风险. 此外, 在 ESDRA 中, 证明设备的新信誉值计算存在高信誉值影响整个结果的风险, 一旦攻击者入侵高信誉值设备, 足以影响整个集群的安全性, 本方案引入中位数, 计算每个证明设备的邻居设备信誉值的中位数, 以此作为基准, 从而规避高信誉值设备“一票否定”的情况, 而在 HEALED 中, 利用传递性, 也同样存在高信任度的设备, 再加上证明节点的任意性, 很难快速找到受损设备.

表 2 不同方案对比

方案	交互模式	拓扑网络	运行时间 (s)	平均能耗 (J)
SEDA	一对多	静态	$t_{\text{total}} \leqslant (448+264H+\sum_{i=0}^{H} m_i)t_{tr}+(1+H)t_{ca}$ $+(2+4H+\sum_{i=0}^{H} m_i)t_h+(1+H)t_p+t_s$	$E \leqslant (80+104m_i)E_s+(104+80m_i)E_r$ $+m_iE_p+(3+3m_i)E_h$
SALAD	一对多	动态	$t_{\text{total}} \leqslant (320+160H)t_{tr}+(1+H)t_{ca}$ $+(1+2H)t_h+(1+H)t_p$	$E \leqslant 100m_i')E_s+(36n+32)m_i'E_r$ $+E_p+2m_i'E_h$
ESDRA	多对一	半动态	$t_{\text{total}} \leqslant ((132+36n)H+\sum_{i=0}^{H} m_i')t_{tr}+(1+H)t_{ca}$ $+(m_i'+2m_i'H+\sum_{i=0}^{H} m_i')t_h+t_p$	$E \leqslant 132E_s+(176+8m_i+136/m_i)E_r$ $+E_p+(5+1/m_i)E_h$
本方案	多对一	半动态	$t_{\text{total}} \leqslant ((32\omega-8)m_i+64H)t_{tr}+m_it_{ca}+m_it_h$ $+m_it_p+(m_i+2)t_s+(m_i+n_i+1)t_v$	$E \leqslant (96\omega-16)E_s+(32\omega+8+(96\omega-16)/m_i)E_r$ $+E_p+(4/m_i)E_h+E_{sg}+E_v$

表 3 安全性比较

安全性	本方案	ESDRA	HEALED
能否准确识别受损设备	可以	可以	不可以
是否考虑物理攻击	考虑	不考虑	不考虑
是否存在会话密钥窃取	不存在	存在	不存在
是否存在高度信任设备	不存在	存在	存在
对受损设备进行的操作	修复	无	修复

(7) 仿真结果. 我们使用 OMNet++[22] 对本文集群证明协议进行了仿真模拟, 并与 ESDRA、SEDA、SALAD 几个代表性工作进行了比较. 在仿真过程中, 我们使用嵌入式开发板上测得的密码算法性能数据[23] 进行仿真, 相关密码算法在嵌入式开发板上的数据如表 4 所示.

表 4 采用的密码操作耗时

密码操作	耗时 (ms)
SM4加密	7.0
SM4解密	7.1
SM2签名	176.0
SM2验证	326.0
SM3节点内存验证速度	43.445 MB/s

对于本方案, 我们设置初始信誉值为 3, 最大信誉值w_{max}为 5, 最小信誉值w_{min}为 1, 调和参数λ为 0.8, 奖励信誉值为 1, 惩罚信誉值为 2. 通过仿真模拟, 以上 4 个方案得出数据如图 4、图 5 所示.

图 4 展现了本方案与 ESDRA、SEDA 以及 SALAD 关于证明设备不同邻居数量的验证时间. 相较于 ESDRA, 由于本方案在上报临时验证结果是采用簇头的公钥进行加密, 并且后期广播证明设备的验证结果采用簇头签名, 因此运行时间有所增加, 但在总体看来, 为了提高协议的安全性, 这些性能牺牲还是在可以接收的范围内, 并且分布式证明的优势依旧可以体现. 图 5 描绘了本方案与 ESDRA、SEDA 以及 SALAD 关于不同的集群设备数量所进行的证明时间, 本方案对于集群数量的增大, 受影响的幅度较小.

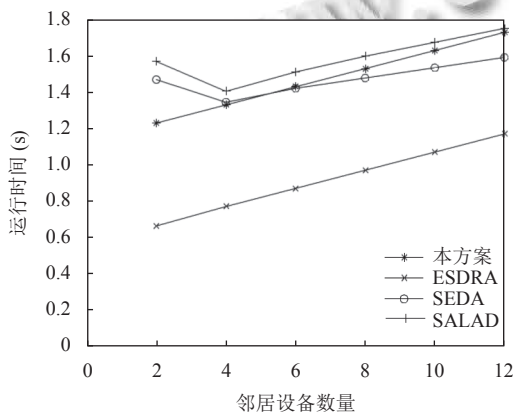

图 4 邻居节点数量与运行时间关系图

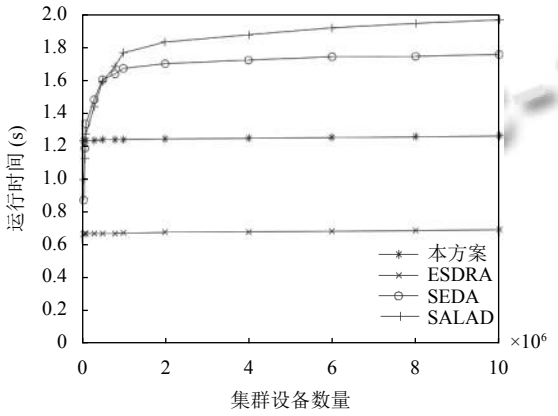

图 5 集群设备数量与运行时间关系图

7 总结

本文提出了一种基于信誉机制和 Merkle 树的安全集群证明及修复方案. 利用信誉机制实现了多对一的证明协议, 不仅能有效解决单点故障, 可以从设备触发验证, 并且适用于半动态网络. 引入 Merkle 树进行度量, 能够快速精确地识别被恶意软件感染的代码块, 并进行高效地恢复到可信状态. 通过对方案的安全性分析和性能评估, 结果表明, 本文集群证明在提高了安全性的同时导致的性能开销是可以接受的.

参考文献

1 Frontera S, Lazzeretti R. Bloom filter based collective remote attestation for dynamic networks. The 16th International Conference on Availability, Reliability and Security. Vienna: ACM, 2021. 80.

2 Neshenko N, Bou-Harb E, Crichigno J, *et al*. Demystifying IoT security: An exhaustive survey on IoT vulnerabilities and a first empirical look on internet-scale IoT exploitations. IEEE Communications Surveys & Tutorials, 2019, 21(3): 2702–2733.

3 Griffioen H, Doerr C. Examining Mirai's battle over the Internet of Things. Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security. ACM, 2020. 743–756. [doi: 10.1145/3372297.3417277]

4 Ibrahim A. Securing embedded networks through secure collective attestation. Proceedings of the 2018 Workshop on MobiSys 2018 Ph.D. Forum. Munich: ACM, 2018. 1–2.

5 Seshadri A, Luk M, Perrig A. SAKE: Software attestation for key establishment in sensor networks. 4th IEEE International Conference on Distributed Computing in Sensor Systems. Berlin: Springer, 2008. 372–385.

6 Li YL, McCune JM, Perrig A. VIPER: Verifying the integrity of peripherals' firmware. Proceedings of the 18th ACM Conference on Computer and Communications Security. Chicago: ACM, 2011. 3–16.

7 McCune JM, Li YL, Qu N, *et al*. TrustVisor: Efficient TCB reduction and attestation. Proceedings of the 2010 IEEE Symposium on Security and Privacy. Oakland: IEEE, 2010. 143–158.

8 McCune JM, Parno BJ, Perrig A, *et al*. Flicker: An execution infrastructure for TCB minimization. ACM SIGOPS Operating Systems Review, 2008, 42(4): 315–328. [doi: 10.1145/1357010.1352625]

9 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.

10 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.

11 Kuang BY, Fu AM, Yu S, *et al*. ESDRA: An efficient and secure distributed remote attestation scheme for IoT swarms. IEEE Internet of Things Journal, 2019, 6(5): 8372–8383. [doi: 10.1109/JIOT.2019.2917223]

12 Ibrahim A, Sadeghi AR, Tsudik G. HEALED: Healing & attestation for low-end embedded devices. 23rd International Conference on Financial Cryptography and Data Security. Cham: Springer, 2019. 627–645.

13 Asokan N, Brasser F, Ibrahim A, *et al*. SEDA: Scalable embedded device attestation. Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security. Denver: ACM, 2015. 964–975.

14 Ibrahim A, Sadeghi AR, Zeitouni S. SeED: Secure non-interactive attestation for embedded devices. Proceedings of the 10th ACM Conference on Security and Privacy in Wireless and Mobile Networks. Boston: ACM, 2017. 64–74.

15 Ambrosin M, Conti M, Lazzeretti R, *et al*. Toward secure and efficient attestation for highly dynamic swarms: Poster. Proceedings of the 10th ACM Conference on Security and Privacy in Wireless and Mobile Networks. Boston: ACM, 2017. 281–282.

16 Kohnhäuser F, Büscher N, Katzenbeisser S. SALAD: Secure and lightweight attestation of highly dynamic and disruptive networks. Proceedings of the 2018 on Asia Conference on Computer and Communications Security. Incheon: ACM, 2018. 329–342.

17 Eldefrawy K, Tsudik G, Francillon A, *et al*. SMART: Secure and minimal architecture for (establishing dynamic) root of trust. Network and Distributed System Security Symposium. San Diego: NDSS, 2012.

18 Zhou L, Su CH, Hu Z, *et al*. Lightweight implementations of NIST P-256 and SM2 ECC on 8-bit resource-constraint embedded device. ACM Transactions on Embedded Computing Systems, 2019, 18(3): 23.

19 Durairaj M, Muthuramalingam K. Dynamic shifting genetic non-adjacent form elliptic curve Diffie-Hellman key exchange procedure for IoT heterogeneous network. 2nd International Conference on Computing and Communication (IC3). Singapore: Springer, 2019. 489–509.

20 武一, 李家兴, 范书瑞, 等. 基于最佳簇半径的无线传感器网络分簇路由算法. 现代电子技术, 2021, 44(4): 23–26.

21 Li FF, Yu XZ, Wu G. Design and implementation of high availability distributed system based on multi-level heartbeat protocol. 2009 IITA International Conference on Control, Automation and Systems Engineering. Zhangjiajie: IEEE, 2009. 83–87.

22 OpenSim Ltd. OMNeT++ discrete event simulator. http://omnetpp.org/. (2016-10-18).

23 杜变霞, 秦宇, 冯伟, 等. 面向物联网的高效集群证明机制. 计算机系统应用, 2018, 27(10): 22–32. [doi: 10.15888/j.cnki.csa.006626]

(校对责编: 牛欣悦)