基于可信执行环境的物联网边缘流处理安全技术综述

姜 超[1], 李玉峰[1,2], 曹晨红[1], 李江涛[1]

[1]上海大学计算机工程与科学学院 上海 中国 200444
[2]网络通信与安全紫金山实验室 南京 中国 211100

摘要 万物互联时代, 物联网中感知设备持续产生大量的敏感数据。实时且安全的数据流处理是面向物联网关键应用中需要解决的一个挑战。在近年兴起的边缘计算模式下, 借助靠近终端的设备执行计算密集型任务与存储大量的终端设备数据, 物联网中数据流处理的安全性和实时性可以得到有效的提升。然而, 在基于边缘的物联网流处理架构下, 数据被暴露在边缘设备易受攻击的软件堆栈中, 从而给边缘带来了新的安全威胁。为此, 文章对基于可信执行环境的物联网边缘流处理安全技术进行研究。从边缘出发, 介绍边缘安全流处理相关背景并探讨边缘安全流处理的具体解决方案, 接着分析主流方案的实验结果, 最后展望未来研究方向。

关键词 物联网; 边缘计算; 可信执行环境; 安全流处理
中图法分类号 TP302 **DOI 号** 10.19363/J.cnki.cn10-1380/tn.2021.05.11

Survey of Security Technologies for IoT Edge Stream Processing Based on Trusted Execution Environment

JIANG Chao[1], LI Yufeng[1,2], CAO Chenhong[1], LI Jiangtao[1]

[1]College of Computer Engineering and Science, Shanghai Univesity, Shanghai 200444, China
[2]Network Communication and Security Purple Mountain Laboratory, Nanjing 211100, China

Abstract In the era of the Internet of Everything, sensing devices in the Internet of Things continue to generate a large amount of sensitive data. The processing of real-time and secure data stream is a challenge that needs to be solved in key applications facing the Internet of Things. In the edge computing model that has emerged in recent years, with the help of devices close to the terminal to perform computationally intensive tasks and store a large amount of terminal device data, the security and real-time performance of data stream processing in the Internet of Things can be effectively improved. However, under the edge-based IoT stream processing architecture, data is exposed to the vulnerable software stack of the edge device, which brings new security threats to the edge. To this end, the article studies the security technology of IoT edge stream processing based on trusted execution environment. Starting from the edge, we first introduce the relevant background of secure edge stream processing, then discuss the specific scheme of edge stream processing and analyze the experimental results of the mainstream scheme, and finally look forward to the future research direction.

Key words Internet of Things, edge computing, trusted execution environment, secure stream processing

1 引言

物联网(Internet of Things, IoT)设备在生活中随处可见, 其形成的一系列物联网智能应用, 如智能运输[1], 智慧城市[2], 智能家居[3]以及智能电网[4]等, 正深刻改变着人们的日常生活以及社会公共服务。据国际著名网络方案供应商思科(Cisco)预测, 2020年将有 500 亿台终端设备接入网络[5], 产生的数据将超过 500ZB[6]。

随着万物互联趋势的不断发展, 越来越多的物联网关键应用需要对感知设备产生的大量数据进行低延时、高吞吐的流式数据分析与处理。例如, 北京和伦敦等大城市已部署了数百万个摄像头进行实时的数据监控与收集, 用于交通的实时监测和控制;

通讯作者: 李玉峰, 博士, 教授, Email: liyufeng_shu@shu.edu.cn; 曹晨红, 博士, 讲师, Email: caoch@shu.edu.cn。
本课题得到广东省重点研发计划(No.2018B010113001), 之江实验室科研项目(先进工业互联网安全平台), 上海市扬帆计划(No.20YF1413700)资助。
收稿日期: 2020-07-24; 修改日期: 2020-12-30; 定稿日期: 2021-03-05

越来越多地建筑物配备了各种各样的传感器, 实时监测照明、 温度、湿度、气压、空气质量等信息, 以提高人员安全性和居住舒适度等。这些数据流中包含了大量的个人位置信息、医疗信息、快递信息、以及政务企业信息等敏感信息, 因此, 对物联网流处理的安全性提出了更高的要求。

传统方法通过对数据进行加密(如 Styx[7], MrCrypt[8])来确保数据的机密性, 而加密操作通常是计算密集型的, 对于资源受限的物联网终端设备来说代价过于昂贵。复杂的加密操作也降低了物联网应用响应的及时性, 影响应用性能。近年来, 很多研究工作利用边缘计算[9-10]构建物联网流处理系统, 在靠近物或数据源头的网络边缘侧, 对数据进行一系列计算、处理及汇总, 从而实现对数据处理的安全性与实时性的提升。典型的物联网边缘流处理基本架构如图 1 所示, 主要分为三个部分, 分别为终端, 边缘端, 云端。物联网数据流由终端流出, 流经边缘进行数据处理与分析, 最后到达云端。例如文献[11]中提到的一个具体案例, 该案例基于上述架构并利用 DNN[12]完成分类任务。其中移动设备终端是数据的来源。边缘对 DNN 的网络层进行划分, 并将计算不密集部分放在边缘, 计算密集型部分放在云端。首先, 移动终端将收集到的等待分类的图片数据发送到边缘; 随后, 边缘将经过部分处理后的结果传到云端; 接着云端对该结果进行验证并进行下一步的处理; 最后云端再将处理后的最终结果传输到移动设备终端。

一方面, 对于传统云计算而言, 边缘更加靠近终端设备, 降低了远距离传输带来的安全风险, 并能够更加及时地响应终端设备。另一方面, 对于终端而言, 边缘拥有更多的计算资源, 数据流处理能力强大。

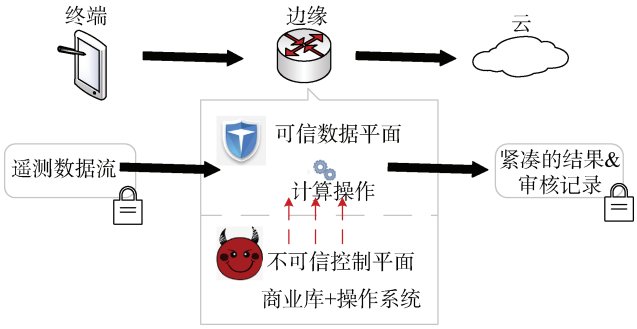

图 1 基于边缘计算的物联网流处理架构

Figure 1 IoT stream processing architecture based on edge computing

目前基于边缘架构的物联网流处理系统依然存在很多安全问题。首先, 与终端设备类似, 边缘设备通常分布式地部署在多种多样的环境中, 缺乏专业有效地监管[13]且配置薄弱[14]。其次, 在边缘侧对物联网数据流的处理往往需要经由一系列复杂的组件, 这些组件存在大量潜在的攻击面与安全威胁。这些组件包括商用操作系统(例如 Linux 或 Windows), 各种用户库以及流分析引擎的运行时框架等等。组件中往往重用了大量为服务器和工作站开发的代码, 其中存在不少可被攻击者利用的配置[15]和漏洞[16]。另外, 边缘处通常汇集了来自多个终端的数据, 是攻击者的高价值目标。与终端相比, 边缘更容易受到攻击, 一旦边缘被攻破, 攻击者不但能访问到机密数据, 而且可能会删除或伪造发送到云端的数据, 从而威胁到整个物联网部署的完整性。如表 1 所示, 总结了典型物联网边缘流处理架构中每个部分的功能、特征、安全技术与安全现状。

表 1 物联网边缘流处理架构各部分功能、特征、安全技术与安全现状

Table 1 The functions, features, security technologies and security status of each part of the edge stream processing architecture of the Internet of Things

名称	功能	特征	安全技术及安全现状
终端	收集数据, 并传输至边缘	计算、存储等资源受限	因资源受限, 许多安全技术无法应用
边缘	终端数据计算, 数据存储, 数据传输, 协助云端进行验证	靠近终端, 时延性较低; 计算资源高于终端, 但低于云端; 硬件设备异构, 协同工作难以保证; 对于数据泄露而言, 边缘的安全性优于终端, 但低于云端	已采用许多传统以及新兴且适合于边缘的安全技术来提高边缘数据流处理的安全性, 但大都存在一定缺陷
云端	SaaS、PaaS、IaaS; 充当私有云或混合云; 大数据分析与存储; 灾难恢复	计算、存储等资源丰富; 距离终端较远, 存在实时性, 带宽不足, 能耗较大等缺点; 可靠性、通用性、可伸缩性、按需服务、廉价、规模巨大、虚拟化	由大型企业进行维护, 其数据流处理安全技术已经十分成熟

为了在边缘平台上进行安全流处理, 本文将探讨基于可信执行环境(Trusted Execution Environment, TEE)的物联网边缘流处理安全技术, 维护物联网数据流的机密性和完整性, 并确保高吞吐量和低输出延迟。其主要思想是在受信任的执行环境中保护和处理数据并限制其访问接口, 忽略其余被认为不受信任的边缘软件堆栈, 从而保证数据流的安全处理。

文章将首先介绍边缘数据流安全处理的相关背

景, 然后对基于可信执行环境相关技术的具体解决方案进行介绍与探讨, 接着对主流解决方案的实验平台与结果进行介绍与分析, 最后对未来边缘数据流安全处理的研究方向进行展望与探讨。

2 边缘安全流处理相关背景

本节将从边缘计算、流处理、可信计算基、可信执行环境四个方面展开, 对边缘安全流处理的相关背景进行介绍。流处理与边缘计算能够解决物联网中数据流的高吞吐量与高延迟的问题。可信执行环境保障边缘数据流的安全处理。可信计算基为可信执行环境构建的基础, 因此, 也为基于可信执行环境的边缘数据流的安全处理机制提供必要的条件。可信执行环境利用可信计算基, 在边缘构造一个安全的环境。在该环境中, 数据的计算、存储、保护, 敏感代码的执行等, 均与不安全的世界隔离, 从而保证边缘数据流的安全处理。另外, 2.4 小节介绍的 RISC 微处理器 TrustZone(Advanced Risc Machines TrustZone, ARM TrustZone)、英特尔软件防护扩展(Intel Software Guard Extensions, Intel SGX)等平台均为支持 TEE 技术的产品。

2.1 边缘计算

边缘计算是一种计算范例, 它利用靠近终端设备的边缘服务器构建一个本地云, 并利用该本地云执行计算密集型任务以及存储大量终端用户数据[17-19]。相比云计算来说, 边缘计算中的本地云更靠近用户终端设备, 能够运行较为复杂的分析, 并得到实时处理结果。边缘计算旨在解决在云计算中未得到适当处理的问题, 如延迟敏感型服务和应用程序中的高延迟问题[20-22]。相对于云计算, 边缘计算在一定程度上提供了优势, 但由于其为新兴领域, 其发展仍处于萌芽阶段。

边缘计算具有如下特征: 边缘设备密集分布, 对移动设备流动性支持, 位置感知, 距离用户近, 网络延迟低, 情境感知, 异构性。表 2 给出了边缘计算特征的具体介绍。

边缘计算的这些独特特征, 能够大大提高物联网应用的性能, 如快速计算物联网终端产生的大量数据流、存储物联网终端数据流及实时响应物联网终端的请求等。文章[23]从网络传输、存储性能及计算能力这三个方面对基于边缘计算的物联网架构带来的优势进行了介绍。而网络传输的快慢, 存储能力的大小, 算力的强弱是决定物联网应用性能高低的关键因素。因此, 近些年来边缘计算在物联网中得到广泛的应用。随着边缘计算的发展, 目前基于边缘的物联网应用也有很多, 并取得了出色的效果, 如公共安全实时数据处理问题[24-26], 智能网联车[27-29], 虚拟现实[30-31]等。

表 2 边缘计算的特征及具体介绍

Table 2 Features and specific introduction of edge computing

特征名	具体介绍
分布密集	边缘计算在终端设备周边部署众多计算平台, 使得云服务更贴近用户
流动性支持	边缘计算支持诸如定位器 ID 分离协议(Locator ID Separation Protocol, LISP)之类的移动性, 以直接与移动设备进行通信
位置感知	边缘计算的位置感知属性, 允许终端设备从最接近其物理位置访问边缘服务
距离近	在边缘计算中, 计算资源和服务距离用户很近, 进而改善用户的使用体验
低延迟	相比云计算, 边缘计算使计算资源和服务更接近终端设备, 从而减少了访问服务的等待时间
情境感知	终端移动设备具有感知上下文的特征, 并与位置感知相互依赖。边缘计算中的移动设备可利用上下文信息做出决策并访问边缘服务
异构性	异构性是指边缘计算元素(终端设备, 边缘服务器等)使用不同的平台, 体系结构, 基础结构, 计算和通信技术

基于边缘计算的物联网架构, 能够对物联网终端设备产生的庞大数据流进行实时的处理以及快速的响应。这种架构利用边缘在资源受限的终端设备与延迟较高的云端之间找到一个平衡点。但由于其发展时间较短, 在应用安全, 数据安全, 网络安全, 以及节点安全等方面仍然面临着巨大的挑战。

2.2 流处理

流处理一般是指对运动中的数据进行处理。换句话说, 就是在生成或接收数据时直接对数据进行处理 [32-34]。物联网中大部分数据都是连续不断的, 如传感器数据, 人的体温、血压等健康信息, 智能网联车的控制信息等。这些数据都是随着时间的流逝而创建的一系列事件。在流处理出现之前, 通常将数据存储在数据库, 文件系统或其他形式的大容量存储中, 并在后期采取批处理的方式对数据进行处理。但对于实时性要求较高的物联网应用来说, 若仍采用这种方式对数据进行处理, 带来的弊端将是灾难性的。

在流处理架构中有一个组件称为流处理器[35-36], 它主要具有两种类型的功能, 分别为用于数据移动和计算的数据功能以及用于资源管理和计算流程的控制功能(例如创建和安排任务)。流处理器基本要求是确保物联网的庞大数据能够在物联网中高效地流动以及被快速及时的计算并拥有一定的容错能力,

保证每个数据至少被处理过一次。目前市面上已有许多成熟的流处理框架，如 Samza[40], Storm[41], Flink[42]等, 但是现存框架难以解决流数据在处理过程中面临的安全挑战。

基于边缘的物联网架构对数据处理的流程如下, 首先物联网产生的数据到达边缘; 随后, 流处理器在管道处对数据进行提取并及进行处理(如计算, 存储, 分组等); 最后, 利用管道将处理结果输出。因此, 物联网中的流数据具有实时性, 易失性, 突发性, 无序性及无限性[37-39]这几个特征。表 3 具体介绍了物联网中的流数据的特征。

表 3 物联网中的数据流的特征及具体介绍

Table 3 The characteristics and specific introduction of data flow in the Internet of Things

特征名	具体介绍
实时性	终端数据是实时产生的, 并要求实时处理。那些潜在价值很大的数据若超时会失去价值
易失性	物联网中的流数据的处理方式是边到达、边处理、边丢弃。极少部分特殊数据会被存储
突发性	数据流来源于不同终端, 这些终端在不同时间和地点的状态不一样, 导致数据流的速率具有突发性
无序性	物联网中多个终端设备充当多个数据源。对于同一个数据源, 终端设备状态会随时间和环境不断变化。从而导致数据流与数据流, 以及数据流内部的数据元素之间都是无序的
无限性	终端设备不停止工作, 新的数据就会不断地产生并累积, 因此流数据的数据量是无限的

有限的终端设备资源无法同时保证数据流处理的安全与实时性要求。即便借用云计算技术, 仍然面临着带宽、能耗的限制, 难以保证物联网流数据的安全与隐私性。此外, 由于云距离终端较远, 传输延时高, 因此, 其实时性难以保障。边缘计算则为该问题的解决带来了新的思路。表 4 介绍了边缘计算为流处理带来的优势与缺点。

表 4 边缘计算在流处理中的优缺点

Table 4 Advantages and disadvantages of edge computing in stream processing

优势	缺点
资源相对终端较为丰富; 在网络边缘处理大量临时数据, 降低网络带宽压力与系统延迟; 将用户数据保存在网络边缘设备上, 减少数据泄露风险	发展时间较短, 在应用安全, 数据安全, 网络安全, 以及节点安全方面等面临着巨大的挑战

边缘计算与流处理的结合能够更好的保证物联网的庞大数据高效流动以及快速及时的计算与容错能力。同时, 对于时延性要求较高的物联网应用也能够及时响应。并且, 能够采用终端设备无法使用的安全机制。

边缘安全流处理目前面临着诸多挑战。在边缘设备资源受限的情况下, 如何高效且快速的处理物联网终端传输的海量数据以及在数据处理过程中如何有效保证数据的安全性。在存在多个终端数据源的情况下, 如何保证多个数据源的数据被安全且快速的处理。在存在多个边缘流处理引擎的情况下, 如何合理利用多个流处理引擎进行数据处理以及保证流数据的处理具有一定的鲁棒性与安全性。

2.3 可信计算基

在数据流处理过程中, 物联网应用系统需要抵御不同类型的攻击、保证系统的稳定运行以及用户的信息安全等。若无法满足以上条件, 可能会面临系统的宕机、用户信息的泄露、物理安全等挑战。可信计算基(Trusted Computing Base, TCB)在一定程度上能够对上述物联网应用系统的能力进行补充。举例来说, 医疗机构的可信计算基通常具有安全机制, 以对其临床信息数据库实施访问控制和用户身份验证, 使攻击者难以获取任何一个患者的信息。为保证边缘数据流的安全处理, 设计一个适用于边缘的安全架构必不可少, 而设计用于这些安全架构的可信计算基更是重中之重。

表 **5** 介绍了 TCB 的定义, 组成部分, 以及功能。另外 TCB 还需要进行固定形式的测试或验证, 只有当测试结果达到一定要求时, 才能将其投入使用。

表 5 TCB 定义、组成部分、功能

Table 5 TCB definition, components and functions

定义	组成部分	功能
计算机系统内保护装置的总体, 包括硬件、固件、软件和负责执行安全策略的组合体	操作系统安全控件; 单个系统硬件; 网络硬件与软件; 安全程序与协议; 本身物理位置	支持用户访问控制与身份验证; 为特定应用程序提供特权与授权; 防止恶意软件渗透以及数据备份

2.4 可信执行环境

2.4.1 可信执行环境的定义

随着物联网应用系统变得越来越复杂, 用户对安全性的要求越来越高。因此, 边缘数据流的安全处理面临了新的挑战。传统的物联网安全技术不再能够满足物联网边缘流处理架构的安全要求, 而可信执行环境的设计能够有效保证物联网数据流的安全处理。

如下为 TEE 的四个不同时期的定义, 这些定义

表明对 TEE 不同的使用和理解:

• Ben Pfaff[44] 认为 TEE 是“专用的封闭虚拟机, 与平台的其余部分隔离。通过硬件内存保护和存储的密码保护, 可以保护其内容免受未经授权的人员的观察和篡改。”

• OMTP[45]认为“TEE 能抵制一组已定义的威胁。它实现与平台的不安全部分的隔离, 对生命周期管理, 安全存储, 加密密钥和应用程序代码保护。”

• GlobalPlatform [46]认为“ TEE 是一个与设备主操作系统同时运行但与设备主操作系统隔离的执行环境。它可以保护其资源免受一般软件攻击。它可以使用多种技术来实现, 并且其安全级别也会相应变化。”

• Jonathan M[47]认为 TEE“旨在实现可信执行的功能集如下: 隔离执行, 安全存储, 远程证明, 安全设置和可信路径。”

从以上定义来看, TEE 可以简单描述为一种安全的, 受完整性保护的处理环境, 并具有内存和存储功能[48], 另外 TCB 是 TEE 设计实现的重要基础。不同于可信平台模块 [49](Trusted Platform Module, TPM), TEE 在帮助系统实现安全的计算、隐私保护和数据保护的同时, 还为第三方提供隔离的执行环境。在该环境中, 代码的执行过程中对 TEE 的拥有的资源(如内存、高速缓存等)具有高度的信任, 并能够保证程序被隔离执行, 数据被安全的存储。另外文献[50]认为 TEE 保证了所执行代码的真实性, 运行时状态(例如 CPU 寄存器, 内存和敏感 I/O)的完整性以及其代码、数据和运行时状态存储在持久性内存中的机密性。此外, 一些基于硬件设计的 TEE 维护特定的凭证表, 云端根据凭证判断边缘设备是否值得信赖。

如表 6 为 TEE 的核心构件方面的定义, 这些构件块的设计能够帮助 TEE 抵御不同攻击。

表 6 TEE 构件块的名称及功能概览

Table 6 Overview of the names and functions of TEE building blocks

名称	功能
安全启动	确保只能加载具有特定属性的代码。如果检测到修改, 引导过程将中断
安全调度	确保 TEE 能够对系统资源实现安全的调度
跨环境交流	定义接口, 允许 TEE 与系统的其余部分通信
安全储存	保证存储数据的机密性, 完整性和新鲜度(防止重放攻击并强制执行状态连续性[51])的存储, 并且只有授权实体才能访问数据
可信的 I / O 路径	保护 TEE 和外围设备(例如键盘或传感器)之间的通信的真实性和机密性。保护输入和输出数据不被恶意应用程序嗅探或篡改

2.4.2 支持可信执行环境的主流硬件平台

1) ARM TrustZone

ARM TrustZone 是物联网网关的典型硬件[52]。大多数现代 ARM 内核都配备了 TrustZone [53], 这是支持 TEE 技术的一种产品。ARM TrustZone 在逻辑上对平台的软硬件资源进行分区, 例如动态随机存储器和 I/O, 使它们分为安全与非不安全的两个世界。安全世界与非安全世界也称为普通操作系统和安全操作系统。在安全世界中执行一些需要保密的操作, 具体如指纹识别、密码处理、安全认证等。其余操作在非安全世界中执行, 具体如用户操作系统、各种应用程序等。安全世界与不安全世界通过一个名为 Monitor Mode 的模式进行转换。这两个世界通过消息传递以及共享内存这两种方式进行通信。另外, 由于许多恶意攻击人员会在系统断电的时候对系统进行攻击, 所以基于 ARM TrustZone 的系统会在引导启动的开始就保证其安全性, 以保证整个系统的安全。表 7 对 ARM TrustZone 的定义, 重要组件以及主要的功能进行介绍。

表 7 ARM TrustZone 的定义, 主要组件以及主要功能

Table 7 Definition, main components and main functions of ARM TrustZone

定义	组件	功能
ARM 针对消费电子设备设计的一种能够抵御各种可能攻击的安全硬件架构	AMVA3 AXI 总线, 安全机制的基础设施; 虚拟化的 ARM Core, 虚拟安全和非安全核; TrustZone 保护控制器 (TrustZone protection controller, TZPC), 控制外设的安全特性; TrustZone 地址空间控制器 (TrustZone Address Space Controller, TZASC)对内存安全与非安全区域划分和保护	保证隔离执行; 保证可信应用的完整; 保证可信数据的机密性; 数据的安全存储与备份

ARM TrustZone 不同于其他 TEE 技术(如 Intel SGX), 其拥有专用的受信任的 I/O。受信任的 I/O 是 ARM TrustZone 的独特功能, 通过 TrustZone TZASC 和 TZPC 这两个硬件组件来实现。另外, 目前 ARM 平台由于其低功耗与出色的性能以及低廉的成本, 十分适合用于边缘的可信执行环境构造, 并保证边缘的安全流处理。

2) Intel SGX

Intel SGX 是 Intel 在 2013 年推出的指令集扩展, 旨在以硬件安全为强制性保障,为用户空间提供可信执行环境。文献[54]对该项技术进行了系统的介绍与分析, Intel 设计一组新的指令集扩展与访问控制机制, 实现不同程序间的隔离运行,使得关键代码和数据的

机密性与完整性免受恶意软件的攻击和泄露。 其主要思想是将合法软件的安全操作封装在一个飞地(enclave)中, 保护其不受恶意软件的攻击。

无论是特权或者非特权的软件(例如操作系统和虚拟机监视程序 [55](Virtual Machine Monitor, VMM))都无法访问飞地, 所以攻击者难以影响飞地中的代码和数据, 并且飞地的安全边界只包含 CPU 和它自身。SGX 创建的飞地也可以理解为一个可信执行环境 TEE, 无论恶意软件的特权等级有多高都无法访问飞地中的数据, 从而保证数据与代码的完整性与机密性。如图 2 所示为 SGX 的核心操作原则, 首先在不可信环境中创建不同的飞地, 随后飞地通过 SGX 提供的调用接口调用可信函数, 数据会在飞地中被处理和分析, 最终返回得到的结果。以上内容均说明 SGX 的设计能够很大程度上提升系统的安全。

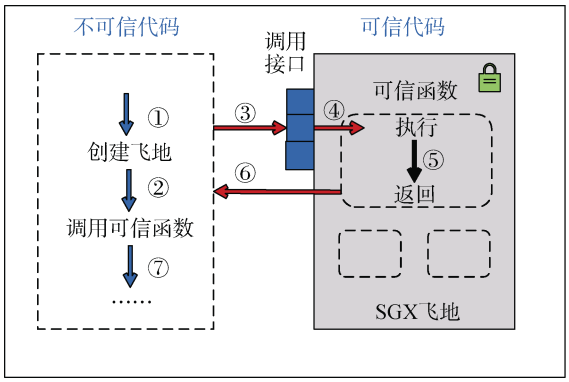

图 2 SGX 核心操作原则

Figure 2 SGX core operating principles

ARM TrustZone 与 Intel SGX 之间的共同点和区别具体如表 8 所示。

表 8 SGX 与 ARM TrustZone 共同点与区别

Table 8 Common points and differences between SGX and ARM TrustZone

名称	共同点	区别
Intel SGX	均支持 TEE 技术; 均适用于边缘云的数据流处理; 均能够提升边缘数据流处理的安全性	SGX 中一个 CPU 可以运行多个安全 enclaves, 且支持并发执行。Enclave 占用的内存会进行硬件加密
ARM TrustZone		TrustZone 中 CPU 在安全世界和正常世界中工作, 两者之间通过 SMC 指令通信

2.4.3 其他支持可信执行环境的硬件平台或相关技术

还有一些支持 TEE 的硬件平台, 如 TI M-Shield[56], Intel TXT[57], AMD SVM[58]也利用隔离的方式来保障系统的安全。TI M-Shield 与 ARM TrustZone 类似, 都适用于嵌入式系统, TI M-Shield 在提供安全的隔离环境的同时, 还提供了安全的移动框架。但是 TI M-Shield 的安全隔离技术只限于两个处理器, 分别为 OMAP 以及 OMAP-Vox[59], 所以应用领域比较狭窄。Intel TXT 和 AMD SVM 都是基于 TPM 发展起来的, 并且提供从 BIOS 到应用层的完整信任链, 从而提高系统的安全性。但由于其对硬件配置及安全控制策略过度依赖导致其无法获得普及。由于这些平台存在一定的缺陷, 目前很少有基于这些平台进行设计的物联网边缘流处理安全方案。

随着物联网应用的不断发展, RISC-V 指令集进入人们的视野。该项目始于 2010 年, 由加州大学伯克利分校的 David A. Patterson 教授主导下完成。RISC-V 是一个全新的指令集架构, 该指令集架构完全开源, 并且允许自由地制造和销售基于该指令集的芯片或软件。相比于其他开源指令集, 其适用于云计算机、手机和嵌入式系统等现代计算设备, 因此具有重大意义。目前市面上常见的基于 RISC-V 设计的系统级芯片(System on Chip, SoC)有 Rocket[60], Hummingbird[61], VexRiscv[62]等。但是由于其诞生时间太短, 相关编译器、软件开发环境和开发工具以及其他生态要素仍然处于发展阶段。目前基于 RISC-V 的边缘安全流处理方案也并不常见。

2.4.4 可信执行环境在终端、边缘以及云端的构建

TEE 的构建能够有效提升物联网应用数据流处理的安全性。但在终端、边缘以及云端构建可信执行环境仍面临诸多挑战, 具体见表 9 所示。

表 9 构建可信执行环境面临的挑战与安全现状

Table 9 Challenges and security status of building a trusted execution environment

名称	挑战	安全现状
终端	计算、存储等资源受限	因资源受限, 可构建的 TEE 方案较为局限
边缘	硬件设备异构, 协同工作难以保证; 对于数据泄露而言, 边缘的安全性优于终端, 但低于云端	传统的一些安全方案也难以应用于边缘。边缘 TEE 方案仍处于发展初期
云端	距离终端较远, 存在实时性, 带宽不足, 能耗较大等缺点。无法满足物联网应用(自动驾驶汽车和虚拟/增强现实)的时延性要求	TEE 技术发展成熟, 但带宽不足与实时性等挑战仍未解决

基于边缘计算的物联网应用架构可以提供较低的延迟, 最大程度地减少传输到后端数据中心所需的流量, 并减少单个云提供商的故障风险。因此, 在

边缘构建 TEE 能够解决终端与云端面临的挑战。

3 边缘安全流处理解决方案

现有的边缘软件堆栈将物联网数据委托给商用操作系统(Operating System, OS), 分析引擎和语言运行环境(例如 Java 虚拟机)。这些组件由成千上万行代码组成[63], 存在许多可利用的漏洞[64-65], 难以保证边缘系统的安全。目前边缘流处理面临的安全问题主要分为两个。一方面, 在边缘端发起攻击的本地攻击者可能会通过广泛的用户/内核接口[66-67]或 IPC [68]攻击分析引擎; 另一方面, 远程的攻击者可能会通过边缘网络服务来损害分析引擎[69]或操作系统[70]。攻击者成功后可能会暴露物联网数据, 破坏或暗中操纵数据。在边缘建立可信执行环境能够有效解决上述问题, 从而保证边缘数据流的安全处理。

3.1 基于软件的可信执行环境技术

3.1.1 基于操作系统进程隔离恶意软件机制

目前许多应用程序对系统拥有极高的特权, 它们能够利用高分辨率相机和其他传感器来观察其用户和物理环境。若这些应用系统被攻击, 很可能造成用户信息的泄露(如银行卡号, 车牌号, 物理位置, 计算机内容)或用户被监视等安全问题。

Jana 等人[71]针对以上问题, 提出了一种隐私保护系统——Darkly。作者重点关注以下场景, 该场景中设备的操作系统与感知传感器的硬件完全可信, 但设备上第三方应用程序正在不可信的环境中运行。系统主要思想为, 即使应用程序是恶意的, 但是只能以用户级特权运行, 并且只能通过可信的 API(例如 OpenCV 计算机视觉库)访问系统(包括感知传感器), 从而提高边缘上数据流处理的安全性。

Darkly 的系统模型如图 3 所示, 其中受信任的组件以阴影表示。Darkly 本身包含两个部分, 受信任的本地服务器和不受信任的客户端库。作者利用操作系统提供的基于用户的标准隔离: 其中 Darkly 服务器是特权进程, 可以直接访问感知传感器, 而应用程序作为非特权进程运行, 只能通过 Darkly 访问传感器。不受信任的 Darkly 客户端库作为每个应用程序进程的一部分运行, 并与 Darkly 服务器通信。即使恶意应用程序修改了相关库, 系统仍保持安全。

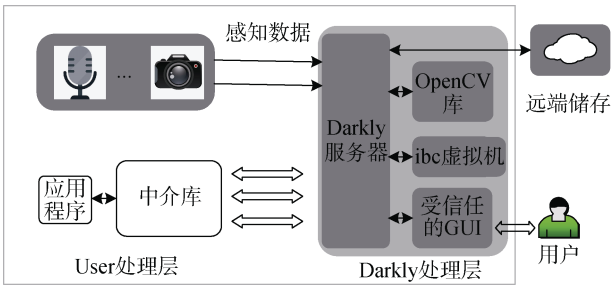

图 3 DARKLY 的系统模型

Figure 3 DARKLY's system model

该方法将数据流的存储、获取、计算等操作与恶意应用程序隔开, 在一定程度上防止数据被恶意程序泄露或者篡改。但其数据的计算主要在 OS 进程中, 这样容易导致 TCB 过大。TCB 越大导致代码量过大, 容易利用的漏洞就越多, 攻击面也越广, TCB 中的任何漏洞都可能使攻击者具有访问应用程序数据或损害其完整性的能力。

3.1.2 基于虚拟化隔离恶意操作系统机制

商用操作系统的可信组件往往包含大量代码, 因此, 具有广泛的攻击面。可信组件一旦被破坏, 攻击者可以完全访问系统上的敏感数据。目前有许多基于虚拟化的安全方案来隔离这些可能受到破坏的 OS, 从而保证应用程序中的敏感数据免受攻击者的读取与修改。Chen 等人[72]提出了 Overshadow, 利用多重阴影技术, 即使整个操作系统都被攻击者破坏, 仍然能够保护应用程序数据的隐私和完整性。Owen S 等人[73]提出的一个基于虚拟化架构的系统—Inktag, 该系统利用安全、可靠的程序对不受信任的商品 OS 的行为进行验证。John 等人[74]提出的 Virtual Ghost 系统, 它能够创建操作系统根本无法读取或修改的安全内存, 从而保护敏感数据的安全。上述安全方案利用虚拟化技术保护应用程序的敏感数据免受感染 OS 内核的侵害。但是, 这样的系统容易受到侧信道攻击。Dong 等人提出了防御一种侧信道攻击的方法, 具体攻击为受到入侵的 OS 内核发起的页表和最后一级缓存(Last Level Cache, LLC)侧信道攻击。

目前基于虚拟化的隔离恶意 OS 的机制在物联网边缘流处理安全方案的应用仍然面临着一些挑战, 主要可以分为三个方面。第一, 上述安全方案大都实现于资源相对丰富的 PC, 并不适用于物联网嵌入式系统; 第二, 上述安全方案并不适用于流处理或无法高效快速的数据流; 第三, 对于边缘设备而言, 上述安全方案代码体量仍然过大, 并具有广泛的攻击面。但虚拟化技术由于其更高的资源利用率、更高的安全性以及更高的可用性和扩展性等优势, 对基于虚拟化的边缘流处理安全方案仍然值得研究。

3.1.3 基于可信计算基的安全机制

为了进一步提升 TCB 的安全性, 缩小 TCB 的尺寸是一种直接可行的措施。只有保持较小的 TCB 安全区域尺寸, 才能够有效保证边缘数据流的安全处理。

文献[75]提出了 Flicker, 一种使用最小 TCB 对敏感代码隔离执行的体系结构。在 Flicker 开始执行之前和执行过程中, 任何软件都无法对 Flicker 代码进行监视或干扰。并且在 Flicker 执行完全结束之前, 可以消除所有 Flicker 代码执行的痕迹。例如, 证书颁发机构(Certificate Authority, CA)可以使用其私钥对证书进行签名, 即使基本输入输出系统(Basic Input Output System, BIOS), OS 和直接存储器访问(Direct Memory Access, DMA)设备被控制, 密钥仍然安全, 并且还提供代码的远程证明。但是由于 Flicker 需要执行 Rootkit[76]检测, 在执行的过程中会进行完整性度量[77]。若完整性度量方法的结果的误报率和漏报率过高, 则可能无法检测出 Rootkit, 从而导致系统被攻击。

文献[78]开发了一个安全的虚拟机管理程序, 称为 TrustVisor, 为安全敏感的代码模块提供安全的执行环境, 另外 TCB 中仅包含 5306 行代码(其中一半以上用于加密操作)。TrustVisor 能够保护不受信任平台上的安全敏感代码和数据, 并阻止恶意软件(例如内核级 Rootkit)的攻击。更具体地说, TrustVisor 旨在保护安全敏感的代码的完整性, 以及该代码所使用数据的机密性和完整性, 并向远程实体证明这些属性。该系统支持许多应用程序(例如安全套接字/安全传输层协议), 特别适用于敏感代码量较小的情况。这项工作的主要贡献是设计和实现了一个综合系统, 使应用程序开发人员能够为在不安全的平台上执行的数据和代码提供强大的安全保证, 并向外部验证者证明这些安全性。同时, TrustVisor 设计了小型的 TCB, 大幅提升了工作效率、易用性和安全性。

以上方法将 TCB 尺寸保持在安全区域内较小, 从而减少 TCB 中存在的漏洞, 使得攻击者难以访问应用程序数据或损害其完整性, 在一定程度上提升了边缘数据流处理的安全性。但在以最小的 TCB 支持数据密集型计算时, 面临诸多挑战。

3.2 基于硬件的可信执行环境技术

3.2.1 基于 Intel SGX 安全流处理方案

攻击者通常攻击现有系统软件中的漏洞, 或者损害特权系统管理员的凭据, 从而获得对其他应用程序数据的访问, 修改或对系统的控制等。SGX 增加了对安全区域的支持, 能够保护应用程序代码和数据免受其他软件(包括特权更高的软件)的访问。SGX 的这个特性能够有效解决上述问题, 并保证边缘数据流处理的安全性。

文献[79]为保证物联网数据流的安全处理, 介绍了一个基于 Intel SGX 的 SecureStreams 中间件框架, 该框架能够支持在不受信任的分布式环境中开发和部署安全流处理应用。其设计将高级反应数据流编程范例与英特尔底层软件防护扩展相结合, 以确保处理数据的私密性和完整性。另外, SecureStreams 支持从大型集群到多租户云基础架构的分布式环境中流处理任务的实现、部署和执行, 能够有效解决数据流处理在分布式环境中面临的安全问题。作者利用 LUA[80](一个小巧的脚本语言)实现了 SecureStreams 的原型。尽管该方案在一定程度上能够保证数据流处理的安全, 但在其 TEE 中缺乏对于数据流并行处理的优化, 可能会导致边缘处理结果的延迟性较高、数据吞吐量较小等问题。

现有的容器隔离机制[81], 仅能防止不受信任的容器对数据的访问, 但无法阻止高级特权系统软件的访问, 如 OS 内核和系统管理程序等。文献[82]针对这一问题, 提出了一种用于 Docker 的安全容器机制-SCONE, 该机制使用 Intel CPU 的 SGX 受信任执行技术来保护数据免受容器进程外的攻击。该系统主要关注以下场景, 一个攻击者拥有像超级用户那样对系统以及物理硬件的访问权, 并能够控制整个软件堆栈, 重播、修改记录以及删除任何网络数据包或对文件系统访问。最终, 该机制能够在不受信任的 OS 上提供并运行安全容器, 从而保证数据的完整性与机密性。但该系统将整个用户应用程序和库放入 TCB 中, 这样会导致 TCB 过大, 致使 TCB 暴露较多的漏洞。

目前云计算基础架构旨在保护特权代码免受不受信任的代码侵害, 无法有效防御云计算中恶意特权软件对系统的攻击(如管理程序和固件), 以及对用户数据的访问。文献[83]针对上述问题, 利用 Intel SGX 提出了一个应用于云端的屏蔽执行系统-Haven, 通过将部分代码和数据放置在 SGX 的安全区中的方式来保证数据免受攻击。该系统利用 SGX 的硬件保护技术来防御特权代码和物理攻击(例如内存探针)。

VC3 [84]针对上述问题提出了一个基于 MapReduce 框架系统, 这是第一个允许用户使用云服务器进行分布式计算的同时, 还能够保持运行代码和数据的机密性, 并确保结果正确性和完整性的系统。即使恶意攻击者可以控制整个云提供商的软件和硬件基础架构, 该系统仍然能够正常的工作, 但计算中若经过认证的物理处理器被攻击, 那将无法保证系统仍然保持正常。系统使用可信的 SGX 处理器[85-86]作为构建模块, 将系统分为受信任和不受信任的部分, 并最大程度地减少其 TCB。该方法主要适用于批处理, 但其设计思想与取得的效果值得参考。

上述的两个系统 VC3 与 Haven 均实现于云平台, 对于资源受限的边缘端未必合适, 但其思想及技术值得运用于边缘安全流处理系统的设计上。尽管该系统在设计 TCB 时遵循最小原则, 但对于边缘安全流处理方案而言仍然较大, 会暴露较多漏洞。另外该系统由于设计与实现上的缺陷, 导致无法检测磁盘块的回滚攻击, 对处理性能有一定的影响。

3.2.2 基于 ARM TrustZone 的安全流处理方案

物联网边缘上的商用 OS, 因代码体量过于庞大导致其面临许多的安全挑战, 而这些设备经常处理机密数据, 攻击者完全有可能利用受感染的 OS 访问这些数据。为了解决这个问题, 传统的方案如在单独虚拟机中处理数据[87], 利用硬件功能或改造商用操作系统, 从而保护应用程序免受恶意操作系统的侵害。但是这些方案并不适用与物联网平台。为此文献[88]针对上述问题开发一个系统——TrustShadow。TrustShadow 将旧版本, 未打补丁的应用程序与受到威胁的 OS 进行隔离。并利用 ARM TrustZone 技术, 将资源划分到安全和不安全的环境中。在安全的世界中, TrustShadow 为安全性至关重要的应用程序构建一个受信任的执行环境。这个受信任的环境由轻量级的系统维护, 并且由该系统协调应用程序与不安全世界中的普通 OS 的通信。作者在具有 ARM TrustZone 的真实芯片板上对 TrustShadow 进行了设计, 并使用微基准测试和实际应用程序对其性能进行了评估, 评估的结果十分出色。但是该系统将整个用户应用程序和库拉到 TCB 中。但正如前文所述, 若 TCB 过大, 则导致流分析引擎及其库很大, 很复杂, 容易受到攻击。

文献[89]基于 ARM TrustZone 提出一个边缘流处理安全引擎——StreamBox-TrustZone, 简称 SBT。其目标是维护物联网数据的机密性和完整性, 支持可验证的结果, 并确保高吞吐量和低输出延迟。该系统遵循最小特权原则, 在 TEE 中保护分析数据和计算并限制它们的接口。该方案能够解决 TCB 过大的问题, 将 TCB 缩小为仅包含受保护的功能、TEE 和安全硬件的部分。该系统的整体框架分为终端、边缘端、云端三个部分。终端是数据流来源, 通过使用安全感测技术保证终端产生可信事件; 边缘端对数据流进行处理和分析操作, 是流处理安全的关键部分; 云端则是受信任的部分, 主要负责对边缘处理结果的验证, 以证明分析结果的正确性。

该引擎利用 ARM TrustZone 构造一个可信执行环境 StreamBox-TrustZone, 将系统软硬件资源划分为安全世界与不安全世界, 从而保证边缘数据流的安全处理。如图 4 所示, SBT 主要分为两个部分, 上半部分是受信任的是数据平面, 下半部分是不受信任的控制平面, 这个部分在流处理过程中会面对许多安全问题。另外终端传送的数据流只会通过 ARM TrustZone 的可信 IO 通道流经安全的数据平面, 并且数据流的所有处理都发生在数据平面中。每当数据流经过数据平面, 数据平面会发送一个特定凭证到控制平面。控制平面会根据需要使用凭证, 调用系统接口安排数据分析管道, 最后数据平面将操作管道划分为多个可信计算原语来对数据流进行处理。处理完成后由数据平面将数据发送到云端进行进一步的验证和分析, 验证数据流的结果正确度与新鲜度。通过创建可信执行环境以及高效的验证机制, 来提高边缘数据流处理的安全性。同时, SBT 还设计特殊的内存管理技术与并行计算技术来提高数据流分析的吞吐量与时延性。

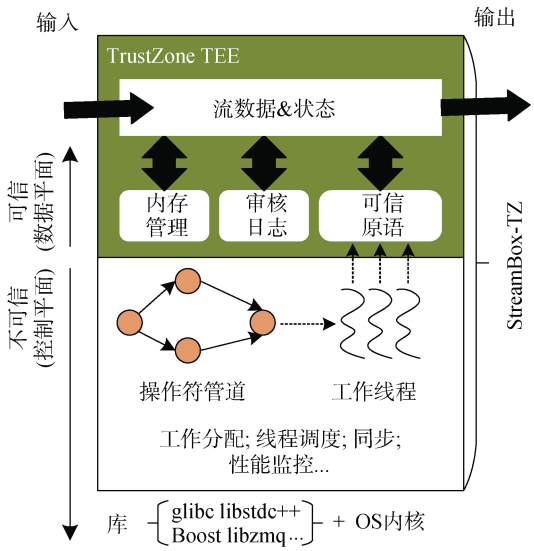

图 4 SBT 框架图

Figure 4 SBT framework diagram

通过创建可信执行环境以及高效的验证机制, 来提高边缘数据流处理的安全性。同时, SBT 还设计特殊的内存管理技术与并行计算技术来提高数据流分析的吞吐量与时延性。Heejin Park 在华为的 Hikey960 开发对系统进行了实现。结果证明, SBT 的 TCB 中的可执行文件只有 42.5 KB, 缩小了 TCB 的大小。在八核 ARMv8 平台上, 它对输入事件处理速度高达 140 MB/s 并具有亚秒级的延迟, 得到了最佳性能。

SBT 为将 TCB 设计到最小, 其仅包含低级计算原语, 导致流管道的机密性无法得到保证。SBT 无法防御以下攻击: (1)可信任的非边缘组件, 如传感器[90];

(2)TEE 内核的错误[91]; (3)侧信道攻击, 如 key skew[92]; (4)物理攻击, 例如嗅探 TEE 的 DRAM[93]。

由于 ARM TrustZone 设计上的不足, 一旦搭配 ARM 平台的终端或者边缘设备丢失, 容易受到内存攻击, 例如冷启动攻击[94], DMA 攻击[95]等。文献[96]针对这些问题提出了 Sentry, 该系统允许应用程序和 OS 组件将代码和数据存储在 ARM SOC 而不是 DRAM 中, 防止攻击者通过内存攻击的方式破坏数据的机密性, 完整性以及系统的可用性。

Sentry 将数据存储在 ARM SOC 上来抵御内存攻击, 但当负责创建和维护 SOC 的操作系统受到威胁时, 所有保存在 ARM SOC 上敏感数据都会泄露出去。文献[97]提出了一个基于 ARM 处理器的缓存辅助安全执行框架-CaSE, 该框架可防御软件攻击, 硬件内存泄露攻击等。CaSE 利用 TrustZone 和 Cache-as-RAM 技术来创建基于缓存的隔离执行环境, 应用程序的数据与代码在内存中进行加密, 并且仅在处理器内解密才能执行。这样能够保护安全敏感型应用程序的代码和数据, 使其免受操作系统和冷启动攻击的侵害。作者在搭配 ARM Cortex-A8 处理器的 IMX53 上实现了 CaSE 的原型。实验结果表明, 在执行包括 AES, RSA 和 SHA1 的加密算法时, CaSE 对系统性能的影响很小。但相比直接处理明文数据, 仍然无法完全满足物联网应用的低延迟要求。

TEE 内核是 SBT 安全世界的重要组成部分, 对处理器拥有最高特权, 并对硬件具有无限访问权。因此, TEE 内核中存在的任何错误都有可能被攻击者利用并攻击, 随后破坏数据的机密性, 安全性以及系统其余部分的正常运行。文献[98]为保证微内核的功能正确性, 提出了 SEL4, 它是 L4 [99]微内核家族的成员, 旨在通过机器辅助和机器检查的形式证明内核的功能正确性。作者实现并证明了 SEL4G 功能的正确性。

3.2.3 基于 RISC-V 的安全流处理方案

当软件开发人员选择目标硬件平台时, 他们的实际应用需求可能会被锁定在不同硬件平台的 TEE 设计限制中。例如, 3.3.1 小节中提到的 Intel SGX 的可信执行环境—enclave 需要的内存大小是静态的, 缺乏安全的 I / O 和系统调用支持, 容易受到侧信道攻击。该情况下, 只有 Intel 才能更改 SGX 中固有的设计, 例如动态调整 SGX enclave 虚拟内存的大小[100]。正是由于这些限制和其他类似限制, 很多开发人员在其他指令集架构(Instruction Set Architecture, ISA)(例如 OpenSPARC[101], RISC-V [102-103])上重新实现 TEE。但对 TEE 进行重新设计时将面临诸多困难, 另外设计出的 TEE 也可能存在上述的限制。为此, Dayeol Lee 提出了 KeyStone[104]—第一个用于构建可定制 TEE 的开源框架。该框架突破固定 TEE 的限制, 允许开发人员构建自定义的 TEE, 从而保证边缘流处理的安全。另外, KeyStone 保证 TEE 原型被快速制作与修复, 对威胁模型具有一定的鲁棒性以及针对特定用例的部署。作者使用 RISC-V 设计和构建 KeyStone, 并在 HiFive Unleashed(搭载 RISC-V 处理器的硬件)上实现了 KeyStone 的原型。其中 RISC-V 是由多个开源核心实现的开放式 ISA [105]。

标记内存是一种提供细粒度和灵活的隔离边界, 从而保证 TEE 的健壮性。其原因有两个, 一方面, 细粒度的隔离边界能够使受信任的内存在有限物理地址中更加紧密集成; 另一方面, 灵活的隔离机制对于动态管理受信任的内存至关重要。但目前标记内存方案产生的内存开销对于资源受限的小型嵌入式系统并不适合。所以, 对于小型嵌入式系统而言, 强大、高效而又灵活的隔离执行仍然是一个未解决的问题。为此 Samuel Weiser 设计了 Timber-V[106], 一种新的带标记的内存体系结构, 该机制能够在小型嵌入式系统上灵活, 高效地隔离代码和数据。作者在 RISC-V 模拟器上将系统实现, 并展示了 Timber-V 的出色性能。作者利用特殊的标记内存技术, 从而保证边缘数据流的安全处理, 该方案将隔离执行机制引入到资源受限的低端设备, 该隔离执行机制类似于 Intel SGX 的 enclave。

TEE 常常无法与外部外围设备建立安全通信, 并且部署在安全环境中, 但该安全环境中的操作系统并不提供最新的防御策略(如保护页面等), 因此, TEE 容易遭受各种攻击。针对上述问题, Pascal Nasahl 提出了 HECTOR-V[107], 一种利用异构多核体系结构开发安全的 TEE 的设计方法。作者认为在主应用处理器上实现的 TEE(例如 Intel SGX 或 ARM TrustZone)是不安全的。因此, 作者另外使用一个安全处理器-RVSCP 将其直接嵌入到 HECTOR-V 架构中。RVSCP 是一种经过安全加固的 RISC-V 处理器, 专为构建 TEE 而设计。该架构在安全域和非安全域之间实现了高度隔离, 并能够抵御侧信道攻击(例如缓存和基于瞬态的攻击)。另外, TEE 与应用处理器的紧密耦合使 HECTOR-V 能够帮助不同设备之间建立安全通信通道。

RISC-V 相较于 Intel SGX 和 ARM TrustZone 起步较晚。因此, 还没有广泛出现于市场的 RISC-V 处理器, 并且对 RISC-V 处理器的隔离执行没有进行广泛的研究。

3.3 边缘安全流处理方案小结

本节将对上述的边缘安全流处理方案进行总结, 主要从三个方面, 分别是可防御攻击、主要方法, 以及主要缺陷。具体如表 10 所示。

表 10 相关技术可防御攻击, 主要方法及主要缺陷

Table 10 Related technologies can defend against attacks, main methods and main flaws

技术名称	主要可防御攻击	主要方法或平台	主要缺陷
MrCrypt, Styx	对数据完整性, 机密性的攻击	数据加密	计算、存储资源消耗大, 难以应用在资源受限的边缘设备上
Darkly	非特权用户攻击	基于 OS 进程隔离	TCB 过大, 系统可利用漏洞过多
Overshadow, Inktag, Virtual Ghost	恶意 OS 的攻击	基于虚拟化技术隔离	实现于 PC 并不适用于物联网设备。并未实现流数据的处理
Flicker, TrustVisor	代码量过多导致的漏洞攻击, 如搜索路径漏洞, Linux 内核漏洞等	缩小 TCB 尺寸	TCB 过小, 无法有效支持数据密集型计算
SecureStreams	数据机密性或完整性攻击, 数据分析正确性的攻击等		缺乏对边缘 TEE 中并行执行的优化, 导致边缘延迟性较高, 数据吞吐量较小等问题
SCONE	恶意高级特权系统或管理软件对软硬件的访问, DRAM 攻击等	基于 SGX 构建一个可信执行环境, 实现与恶意代码的隔离	TCB 过大, 系统可利用漏洞过多
Haven	特权软件攻击		无法检测回滚攻击, 另外适用于云平台
VC3	内部恶意人员攻击		无法防御拒绝服务, 辅助渠道和流量分析攻击, 另外使用于云平台且适用于批处理
StreamBox-TrustZone	破坏数据的机密性或完整性, 破坏分析的正确性, 攻击系统分析性能或可用性等	基于 ARM TrustZone 构建可信执行环境, 缩小 TCB, 优化内存管理	无法防御物理攻击, 侧信道攻击, TEE 内核攻击等
TrustShadow	保护应用程序免受可利用的错误或因配置错误影响的操作系统的侵害		TCB 过大, 系统可利用漏洞过多
KeyStone, Timber-V, HECTOR-V	防止恶意 OS 的攻击	基于 RSIC 构建一个可信执行环境, 实现与恶意代码的隔离	诞生较晚, 相关编译器、软件开发环境和开发工具以及其他生态要素仍然处于发展阶段
Sentry	冷启动攻击, DMA 攻击等	将敏感代码及数据存储在 ARM SOC 上	若管理 ARM SOC 的操作系统被攻击, 保存的数据易泄露
CaSE	操作系统攻击, 冷启动攻击等	利用 AMR TrustZone 与 Cache-as-RAM 来创建可信执行环境	对数据进行加密解密, 无法满足物联网应用对低时延的高要求
SEL4	TEE 内核漏洞攻击	机器辅助和机器检查	暂无系统设计应用于边缘设备

以上部分方案并不完全适用于边缘或边缘安全流处理, 但其设计思想值得在边缘应用, 如基于虚拟化技术隔离技术、Haven 和 SEL4 等。因此, 在设计边缘安全流处理方案时, 上述方案均值得研究。

目前适用于边缘的安全方案大都存在一些缺陷。具体如下, 加密的方法会消耗过大的资源且数据处理速度会大幅下降, 难以在边缘应用。基于 OS 进程隔离机制会导致过大的 TCB, 容易暴露庞大的攻击面。利用缩小 TCB 的方法, 无法有效支持数据密集型计算, 难以满足物联网庞大数据流的快速处理。基于 Intel SGX 的方案成本较高且能耗较高, 另外大都适用于云端。基于 RISC-V 的方案目前仍处于初期, 但其完全开源且适用于终端、边缘和云端, 具有广阔的发展前景。TrustZone 具有低能耗, 出色的性能, 低廉的成本和成熟的技术等特性。因此, 基于 TrustZone 的安全方案目前最适用于物联网边缘。但由于其设计缺陷, 无法防御物理攻击, 侧信道攻击, TEE 内核攻击等。

4 硬件辅助的边缘典型流处理安全方案实验结果对比分析

为应对边缘流处理过程中面临的安全威胁, 通常, 可以采用基于软件的 TEE 安全方案来进行防护, 如 Darkly, 密码加密, 访问控制策略等机制。但基于软件的安全方案性能开销巨大, 对于资源受限的边

缘来说并不适用。另外, 单纯基于软件的 TEE 安全方案, 代码行数可能达到百万级别。如 3.1 小节所述, 代码行数越多, 攻击面越大。而硬件辅助的 TEE 安全方案能够有效解决上述问题。本节将对 2.4 小节介绍的支持 TEE 的硬件平台进行分析与总结, 并对两个基于主流硬件平台的安全流处理方案的具体实验结果进行分析与对比。

4.1 硬件平台分析与总结

2.4.2 小节中说明目前常见的支持 TEE 的硬件平台有 ARM Trustzone, Texas Intruments M-Sheild, Intel SGX 以及 AMD SVM 等。该小节说明 Texas Intruments M-Sheild, AMD SVM 等平台存在一定的缺陷(如应用领域狭窄, 难以普及等), 因此, 并不适用于物联网边缘安全流处理这一场景。由于 RISC-V 指令集架构诞生时间较短, 基于 RISC-V 设计的安全流处理方案并未得到广泛研究。

ARM TrustZone 与 Intel SGX 相比于其他硬件平台更加适用于边缘可信执行环境的构建, 并保证物联网流数据的安全处理。一方面, 由于 Intel SGX 健壮、可信、灵活的安全功能与硬件扩展的性能保证, 使得这项技术在边缘安全流处理方面具有广阔的应用空间与发展前景。目前学术界和工业界已经对 SGX 技术展开了广泛的研究, Intel 也在其最新的第六代 CPU 中加入了对 SGX 的支持, 并加紧研究适用于边缘的 Intel 处理器。另一方面, 根据 2018 年的 Eclipse 一项调查[52], 边缘平台一般使用 ARM TrustZone 来构建边缘安全系统。因此, ARM TrustZone 和 Intel SGX 是目前适用于边缘构建安全流处理方案的主流硬件平台。

表 11 对这些支持 TEE 的不同的平台进行了总结。

4.2 主流硬件平台安全方案实验结果分析

4.1 小节说明 Intel SGX 和 ARM TrustZone 目前市场上的两个主流硬件平台。SecureStreams 和 StreamBox-TrustZone 在流数据处理方面的表现都十分出色, 并且实验配置, 评判基准, 以及安全性也都近似。因此, 本节将对这两个方案实验结果进行分析与对比, 前者基于 Intel SGX, 后者基于 ARM TrustZone。

4.2.1 StreamBox-TrustZone 实验结果分析

SBT 利用缩小 TCB 尺寸, 优化 TEE 内部计算与内存管理机制以及压缩审计记录等技术来保证处理终端流数据的速度与安全性。作者在华为的开发板-Hikey960 上对 SBT 进行了实现与评估。Hikey 板支持 OP-TEE[108]的运行, 并搭载 ARM TrustZone 以及支持第三方编程。该板具体配置见表 12。

表 11 支持 TEE 的硬件平台

Table 11 Hardware platforms supporting TEE

名称	适用领域	主要思想	存在缺陷
ARM TrustZone	嵌入式	利用虚拟化将物理处理器划分为安全与不安全处理器	无法防御物理攻击, 侧信道攻击, TEE 内核攻击等
TI M-Shield		定义安全的 ROM 与 RAM 并嵌入一个安全状态机	过度依赖于硬件配置及安全控制策略, 导致其应用领域比较狭窄
Intel TXT 和 AMD SVM	PC 端	扩展 x86 的指令集	普及度不高
Intel SGX		逆向沙盒	旧版应用程序不容易迁移到 SGX
Rocket, Hummingbird, VexRiscv	嵌入式, PC 端	基于 RISC-V 指令集架构自定义构建 TEE 安全方案	诞生时间较晚, 发展尚未成熟

表 12 SBT 开发板配置

Table 12 SBT development board configuration

SoC	HiSilicon Kirin 620, TDP 36W	CPU	8x ARM Cortex-A53@1.2 GHz
Mem	2GB LPDDR3@800 MHz	OS	Normal: Debian 8 (Linux 4.4) Secure: OP-TEE 2.3

作者设定了相同 CPU 内核数量与输出延迟, 并测试了 SBT(在 TEE 内处理数据, 对数据加密, 并利用可信 I/O)以及三个修改版的 SBT。这三个 SBT 修改版分别是 SBT ClearIngress(在 TEE 内处理数据, 以明文方式提取数据), SBT IOviaOS(在 TEE 内处理数据, 并不利用可信 I/O), Incure(在 TEE 外处理数据, 没有任何安全措施)。另外, 作者使用六个行为基准来判断 SBT 处理终端数据流的能力, 这些基准均包含主要的流操作符。这六个基准具体如表 13 所示。

表 13 基准的名称及具体含义

Table 13 The name and specific meaning of the benchmark

名称	具体含义
Top Values Per Key	根据键值对事件进行分组, 并在每个窗口的每个组中标识 K 个最大值
Counting Unique Taxis	对唯一的出租车 ID 标识, 并在每个窗口中对其进行计数
Temporal Join	连接两个输入流进入相同窗口中具有相同键值的事件
Windowed Aggregation	汇总每个窗口输入值得数量
Filtering	过滤不属于某个窗口范围值的数据
Power Grid	找出了带有大功率插头的房屋

作者测试了所有基准的吞吐量, 每个基准的测试结果都类似, 我们以 Windowed Aggregation 基准为例, 介绍在不同内核数下的不同 SBT 的吞吐量大小, 具体如表 14 所示。

表 14 不同 SBT Windowed Aggregation 的吞吐量结果

Table 14 Throughput results of different SBT Windowed Aggregation (MB/s)

名称	4 核, 延迟 20ms	8 核, 延迟 20ms
Inscure	230	200
SBT ClearIngress	180	180
StreamBox-TZ	65	140
SBT IOviaOS	60	120

最终结果表明, Inscure 表现最佳, SBT ClearIngress 第二, StreamBox-TZ 其次, SBT IOviaOS 相对最差。但是 Inscure 安全性最差, 而 SBT 的安全性是最好的, 并且随着内核数的增加, SBT 的吞吐量也在不断增加。

4.2.2 SecureStream 实验结果分析

SecureStreams 主要用于部署和处理大规模的安全流, 而目前基于 Intel SGX 部署的边缘安全方案很少, 适用于流处理的安全方案更是少之又少。SecureStreams 使用两台计算机组成集群来进行实验与评估, 具体配置如表 15 所示。

表 15 SecureStreams 开发板实验配置

Table 15 SecureStreams experimental configuration

CPU	Intel®Core™i7-6700@3.4GHz, TDP 130W	Mem	8 GB RAM
OS	Normal: Debian 8 (Linux 4.4) Secure: OP-TEE 2.3		

为了衡量系统可实现的吞吐量以及体系结构的网络开销, 作者在三种不同的配置中部署了 SecureStreams 管道。第一个配置是允许流框架绕过 SGX 飞地对数据直接进行处理, 并且不对数据集加密。第二个配置是对数据集进行加密, 但允许加密的数据不在 SGX 飞地中进行处理。最后, 配置一个完全安全的管道, 对输入数据集进行加密, 并在 SGX 飞地中对进行数据处理。数据节点不断的注入数据集, 辅助组件通过非阻塞 I / O 连续侦听传入的数据, 并对数据进行处理。作者通过利用 Docker 的内部监视和统计模块来收集带宽测量结果, 具体结果(以每种配置吞吐量峰值为例)见表 16。

表 16 不同 SecureStreams 配置的吞吐量结果

Table 16 Throughput results of different SecureStreams configurations (MB/s)

名称	1 个辅助组件	2 个辅助组件	3 个辅助组件
配置 1	12	7	3
配置 2	22	13	6
配置 3	23	25	8

最终结果表明, 安全机制越多, 数据吞吐量越小。而数据的处理速度会随着辅助组件的增加而增加, 但是加速度会随着辅助组件增加而变小。其中的原因是辅助组件能够并行处理数据, 而辅助组件的并行会受到物理核心总数的限制。

4.2.3 StreamBox-TrustZone 与 SecureStream 结果分析与对比

从上述两个实验结果可以看出, 在相同实验配置的情况下(均进行数据加密, 均 TEE 内进行数据处理, 相同数量的并行组件处理), SBT 每秒处理的数据更多, 性能更加出色。比如在 4 核心情况下, SBT 的吞吐量能够达到 65MB/s 左右, 而 SecureStream 只能够达到 8MB/s 左右。

SecureStreams 在小型 x86 集群实现了该系统, 该集群的资源比 HiKey 丰富得多, 其具有更快的 CPU(8 倍 i7-6700@3.4GHz 与 8 倍 Cortex-A53@1.2GHz), 更大的 DRAM(16 GB 对 2 GB), 更高的功率(130W 对 36W)和更高的成本(600 美元对 65 美元)。因此 SBT 利用更小的成本以及计算资源得到比 SecureStreams 更好的实验效果。

具体实验结果如表 17 所示。

表 17 主流边缘流处理方案实验结果

Table 17 Experimental results of mainstream edge stream processing scheme

名称	实验平台	实验结果
SecureStreams	Intel SGX	在 Intel®Core™i7-6700 处理器上对于非加密数据平均处理速度能够达到 7.5MB/s, 最高达到 12MB/s, 对加密数据的处理, 每秒的吞吐量很少达到 5MB
StreamBox-TrustZone	ARM TrustZone	在八核 ARMv8 平台上, 处理速度高达 140 MB/s 并具有亚秒级的延迟。另外 SBT 的安全机制所产生的间接费用不到 25%

综上所述, 在边缘安全数据流处理过程中, SBT 能够以更小的计算资源及更低的成本来获得更出色的性能, 但 SecureStreams 能够防御一些 SBT 无法防御的攻击, 如冷启动攻击, DMA 攻击等。

5 安全展望

在之前的章节中, 对基于可信执行环境的物联网边缘流处理安全技术进行了介绍与探讨。但目前对物联网边缘流处理安全的研究还处于一个早期阶段, 依然存在着一系列的安全问题尚待解决, 在这里我们对未来研究方向进行讨论和展望。

(1)边缘数据流处理中的安全与效率的权衡问题。大多数的物联网应用场景需要边缘节点进行复杂的运算操作, 这些复杂的操作往往会消耗很多的时间以及大量的计算资源。若边缘设备上采用的安全机制所消耗的计算资源以及时间代价太高, 会加重物联网设备与边缘之间的延迟问题, 所以在物联网的实际应用场景中需要对安全和效率之间做好权衡, 在保证边缘数据处理安全的情况下同时也要能够保证能够及时对终端设备进行响应。

(2)跨越多个边缘设备的流处理安全问题。在物联网边缘端设备往往不只一个, 可能是一个或者多个边缘设备共同合作进行流处理操作。边缘端的设备一般都是分布式的, 其中网络设备的可信度仍然需要进行进一步的研究, 从而提出一种有效的信任模型来保证跨越多个边缘设备的数据流处理的安全性。

(3)低功耗的高效加密措施。目前, 基于数据加密的方法难以应用于边缘流处理的场景, 主要原因在于 1)边缘设备的资源受限; 2)加解密算法过于复杂; 3)缺乏针对边缘侧的数据加密处理方案及标准。

(4)ARM TrustZone 的安全问题。由于 ARM TrustZone 设计上的缺陷, 目前基于 ARM TrustZone 构建的边缘流分析引擎均难以防御 TEE 内核缺陷攻击、侧信道攻击和物理攻击(内存攻击)等攻击。尽管有一些方法能够在一定程度上防御以上攻击, 但这些方法还未应用于基于 ARM TrustZone 平台的边缘安全流处理引擎。未来可以借鉴并利用这些方法, 来设计基于 ARM TrustZone 平台的边缘安全流处理引擎, 从而解决上述问题, 提高边缘流处理的安全性。

(5)基于 RISC-V 的物联网边缘流处理安全方案的研究。目前市面上的 TEE 硬件平台普遍存在一些问题。例如, Intel SGX 成本太高暂时无法广泛适用于物联网; ARM TrustZone 是封闭式的指令集架构且存在着高昂的专利和架构授权问题; TI M-Shield 过度依赖于硬件配置及安全控制策略, 导致其应用领域比较狭窄。RISC-V 适用于云计算机、手机和嵌入式系统等现代计算设备, 并且完全开源, 因此, RISC-V 前景十分广阔。在未来的物联网市场, RISC-V 凭借其完全开源的优势, 很可能击败 ARM。因此, 基于 RISC-V 的物联网边缘安全流处理方案值得研究。

6 结束语

随着物联网的普及, 越来越多的感知设备每时每刻都产生着大量数据, 需要进行低延时、高吞吐的流式数据分析与处理。在这些数据流中往往包含了大量涉及用户隐私的敏感数据, 在要求实时性的同时, 对流处理的安全性提出了更高的要求。边缘计算的出现一定程度上提升了数据流处理的安全性与实时性然而, 基于边缘计算的架构会使数据暴露于边缘端结构复杂、易受攻击的软件堆栈中, 从而带来诸多的安全威胁。为了解决这个问题, 本文探讨了基于可信执行环境的物联网边缘流处理安全技术, 从边缘安全流处理相关背景出发, 介绍了流处理, 边缘计算以及适用于边缘的可信执行环境技术与相关平台; 然后, 介绍并探讨了用于边缘端的基于可信执行环境的流处理安全技术; 接着, 对主流解决方案的实验平台与结果进行介绍与分析; 最后, 对未来物联网边缘安全流处理的研究方向进行了展望。

参考文献

[1] Greengard S. Smart Transportation Networks Drive Gains[J]. *Communications of the ACM*, 2015, 58(1): 25-27.

[2] Pan G, Qi G D, Zhang W S, et al. Trace Analysis and Mining for Smart Cities: Issues, Methods, and Applications[J]. *IEEE Communications Magazine*, 2013, 51(6): 120-126.

[3] GhaffarianHoseini A, Dahlan N D, Berardi U, et al. The Essence of Future Smart Houses: From Embedding ICT to Adapting to Sustainability Principles[J]. *Renewable and Sustainable Energy Reviews*, 2013, 24: 593-607.

[4] Massoud Amin S, Wollenberg B F. Toward a Smart Grid: Power Delivery for the 21st Century[J]. *IEEE Power and Energy Magazine*, 2005, 3(5): 34-41.

[5] Evans D. The internet of things: How the next evolution of the internet is changing everything[J]. *Cisco Public*, 2011, 1(2011): 1-11.

[6] Networking C V. Cisco global cloud index: Forecast and metho-dology, 2015-2020.white paper[J]. *Cisco Public*, San Jose, 2016.

[7] Stephen J J, Savvides S, Sundaram V, et al. STYX: Stream Processing with Trustworthy Cloud-Based Execution[C]. *SoCC '16: Proceedings of the Seventh ACM Symposium on Cloud Computing*. 2016: 348-360.

[8] Tetali S D, Lesani M, Majumdar R, et al. MrCrypt: Static Analysis for Secure Cloud Computations[C]. *The 2013 ACM SIGPLAN international conference on Object oriented programming systems languages & applications*, 2013: 271-286.

[9] Satyanarayanan M. The Emergence of Edge Computing[J]. *Computer*, 2017, 50(1): 30-39.

[10] Shi W S, Cao J, Zhang Q, et al. Edge Computing: Vision and Challenges[J]. *IEEE Internet of Things Journal*, 2016, 3(5): 637-646.

[11] Li E, Zhou Z, Chen X. Edge Intelligence: On-Demand Deep Learning Model Co-Inference with Device-Edge Synergy[C]. *The 2018 Workshop on Mobile Edge Communications*, 2018: 31-36.

[12] Mikolov T, Sutskever I, Chen K, et al. Distributed Representations of Words and Phrases and Their Compositionality[EB/OL]. 2013: arXiv: 1310.4546[cs.CL]. https: //arxiv.org/abs/1310.4546

[13] Manyika J. The Internet of Things: Mapping the value beyond the hype[M]. McKinsey Global Institute, 2015: 3.

[14] Kashyap A, Horbury A, Catacutan A. Internet Security Threat Report 2014. http://www.symantec.com/content/en/us/enterprise/ot-her_resources/bistr_main_report_v19_21291018.en-us.pdf. 2017.

[15] Yin Z N, Ma X, Zheng J, et al. An Empirical Study on Configuration Errors in Commercial and Open Source Systems[C]. *The Twenty-Third ACM Symposium on Operating Systems Principles - SOSP '11*, 2011: 159-172.

[16] Tan L, Liu C, Li Z M, et al. Bug Characteristics in Open Source Software[J]. *Empirical Software Engineering*, 2014, 19(6): 1665-1705.

[17] Khan W Z, Ahmed E, Hakak S, et al. Edge Computing: A Survey[J]. *Future Generation Computer Systems*, 2019, 97: 219-235.

[18] Hassan N, Gillani S, Ahmed E, et al. The Role of Edge Computing in Internet of Things[J]. *IEEE Communications Magazine*, 2018, 56(11): 110-115.

[19] Ahmed E, Ahmed A, Yaqoob I, et al. Bringing Computation Closer Toward the User Network: Is Edge Computing the Solution[J]. *IEEE Communications Magazine*, 2017, 55(11): 138-144.

[20] Mao Y Y, You C S, Zhang J, et al. A Survey on Mobile Edge Computing: The Communication Perspective[J]. *IEEE Communications Surveys & Tutorials*, 2017, 19(4): 2322-2358.

[21] Constant N, Borthakur D, Abtahi M, et al. Fog-assisted wiot: Asmart fog gateway for end-to-end analytics in wearable internet of things[EB/OL]. 2017: arXiv preprint arXiv: 1701.08680.

[22] Griffey J. 3-D Printers for Libraries[J]. *Library Technology Rep-orts*, 2017, 53(5): 1-31.

[23] Yu W, Liang F, He X F, et al. A Survey on the Edge Computing for the Internet of Things[J]. *IEEE Access*, 2018, 6: 6900-6919.

[24] Zhang Q Y, Yu Z F, Shi W S, et al. Demo Abstract: EVAPS Video Analysis for Public Safety[C]. *2016 IEEE/ACM Symposium on Edge Computing*, 2016: 121-122.

[25] Sun H, Liang X, Shi W S. VU: Video Usefulness and Its Application in Large-Scale Video Surveillance Systems: An Early Experience[C]. *The Workshop on Smart Internet of Things - SmartIoT '17*, 2017: 1-6.

[26] Wu X P, Dunne R, Zhang Q Y, et al. Edge Computing Enabled Smart Firefighting: Opportunities and Challenges[C]. *The fifth ACM/IEEE Workshop on Hot Topics in Web Systems and Technologies - HotWeb '17*, 2017: 1-6.

[27] Gerla M, Lee E K, Pau G, et al. Internet of Vehicles: From Intelligent Grid to Autonomous Cars and Vehicular Clouds[C]. *2014 IEEE World Forum on Internet of Things*, 2014: 241-246.

[28] Dimitrakopoulos G, Demestichas P. Intelligent Transportation Systems[J]. *IEEE Vehicular Technology Magazine*, 2010, 5(1): 77-84.

[29] Zhang J P, Wang F Y, Wang K F, et al. Data-Driven Intelligent Transportation Systems: A Survey[J]. *IEEE Transactions on Intelligent Transportation Systems*, 2011, 12(4): 1624-1639.

[30] Li Y, Gao W. MUVR: Supporting Multi-User Mobile Virtual Reality with Resource Constrained Edge Cloud[C]. *2018 IEEE/ACM Symposium on Edge Computing*, 2018: 1-16.

[31] Lai Z, Hu Y C, Cui Y, et al. Furion: Engineering high-quality immersive virtual reality on today's mobile devices[C]. *the 23rd Annual International Conference on Mobile Computing and Net-working (Mobicom)*, 2017: 409-421.

[32] Akidau T, Schmidt E, Whittle S, et al. The Dataflow Model: A Practical Approach to Balancing Correctness, Latency, and Cost in Massive-Scale, Unbounded, Out-of-Order Data Processing[J]. *The VLDB Endowment*, 2015, 8(12): 1792-1803.

[33] Lin W, Qian Z, Xu J, et al. Streamscope: continuous reliable di-stributed processing of big data streams[C]. *13th USENIX Symp-osium on Networked Systems Design and Implementation,*2016: 439-453.

[34] Murray D G, McSherry F, Isaacs R, et al. Naiad: A Timely Dataflow System[C]. *The Twenty-Fourth ACM Symposium on Operating Systems Principles*, 2013: 439-455.

[35] Miao H, Park H, Jeon M, et al. Streambox: Modern stream pr-ocessing on a multicore machine[C]. *2017 USENIX Annual Tec-hnical Conference*, 2017: 617-629.

[36] R. Clapis. Go get my/vulnerabilities: an in-depth anal-ysis of go language. https://www.blackhat.com/docs/asia-17/materials/asia-17-Clapis-Go-Get-My-Vulnerabilities-An-In-Depth-Analysis-Of-Go-Language-Runtime-And-The-New-Class-Of-Vulnerabilities-It-Introduces.pdf. 2017.

[37] Qin X P, Wang H J, Du X Y, et al. Big Data Analysis—Competition and Symbiosis of RDBMS and MapReduce[J]. *Journal of Software*, 2012, 23(1): 32-45.
(覃雄派, 王会举, 杜小勇, 等. 大数据分析——RDBMS 与 MapReduce 的竞争与共生[J]. *软件学报*, 2012, 23(1): 32-45.)

[38] Hoi S C H, Wang J L, Zhao P L, et al. Online Feature Selection for Mining Big Data[C]. *The 1st International Workshop on Big Data, Streams and Heterogeneous Source Mining Algorithms, Systems,*

Programming Models and Applications - BigMine '12, 2012: 93-100.

[39] Michael K, Miller K W. Big data: New opportunities and new challenges(guest editors' introduction) [J]. *Computer*, 2013, 46(6): 22-24.

[40] Noghabi S A, Paramasivam K, Pan Y, et al. Samza[J]. *The VLDB Endowment*, 2017, 10(12): 1634-1645.

[41] Toshniwal A, Taneja S, Shukla A, et al. Storm@twitter[C]. *The 2014 ACM SIGMOD International Conference on Management of Data*, 2014: 147-156.

[42] Carbone P, Katsifodimos A, Ewen S, et al. Apache flink: Strea-m and batch processing in a single engine[J]. *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering*,2015, 36(4): 28-38.

[43] Lu G, Zeng W H. Cloud Computing Survey[J]. *Applied Mechanics and Materials*, 2014, 530/531: 650-661.

[44] Garfinkel T, Pfaff B, Chow J, et al. Terra: A Virtual Machine-Based Platform for Trusted Computing[C]. *The nineteenth ACM symposium on Operating systems principles - SOSP '03*, 2003: 193-206.

[45] Open Mobile Terminal Platform Consortium. OMTP advanced t-rusted environment: OMTP TR1 (v1.1). http://www.omtp.org/ Pub-lications.aspx. 2009.

[46] Global Platform. TEE system architecture. http://www.globalplatf-orm.org/specificationsdevice.asp. 2011.

[47] Vasudevan A, Owusu E, Zhou Z, et al. Trustworthy execution on mobile devices: What security properties can my mobile pla-tform give me[C]. *The 5th International Conference on Trust and Trustworthy Computing,* 2012: 159-178.

[48] Asokan N, Ekberg J E, Kostiainen K, et al. Mobile Trusted Computing[J]. *The IEEE*, 2014, 102(8): 1189-1206.

[49] Morris T. Trusted Platform Module[M]. Encyclopedia of Cryptography and Security. Boston, MA: Springer US, 2011: 1332-1335.

[50] Sabt M, Achemlal M, Bouabdallah A. Trusted Execution Environment: What it Is, and what it is not[C]. *2015 IEEE Trustcom/BigDataSE/ISPA*, 2015: 57-64.

[51] Parno B, Lorch J R, Douceur J R, et al. Memoir: Practical State Continuity for Protected Modules[C]. *2011 IEEE Symposium on Security and Privacy*, 2011: 379-394.

[52] Eclipse Foundation. Key Trends from the IoT Developer Survey.https://iot.eclipse.org/resources/iot-developer-survey/iot-devel oper-su-rvey-2018.pdf. 2018.

[53] Arm TrustZone. http://www.arm.com/products/processors/ tech-nolog-ies/trustzone/index.php.

[54] Wang J, Fan C Y, Cheng Y Q, et al．Analysis and Research onSGX Technology[J]. *Journal of Software*, 2018, 29(9): 2778-2798.

[55] Agesen O, Garthwaite A, Sheldon J, et al. The Evolution of an X86 Virtual Machine Monitor[J]. *ACM SIGOPS Operating Systems Review*, 2010, 44(4): 3-18.

[56] Srage J, Azema J. M-Shield mobile security technology. http://f-ocus.ti.com/pdfs/wtbu/ti m-shield whitepaper.pdf. 2005.

[57] Greene J. Intel Trusted Execution Technology-Hardware-based Technology for Enhancing Server Platform Security[J]. *Intel Corporation,* 2010, 2012(8): 2015.

[58] Virtualization A. Secure virtual machine architecture reference manual. http://www.0x04.net/doc/amd/33047.pdf. 2005.

[59] Cumming P. The TI OMAP™ Platform Approach to SOC[M]. Winning the SoC Revolution. Boston, MA: Springer US, 2003: 97-118.

[60] Asanovic K, Avizienis R, Bachrach J, et al. The rocket chip ge-nerator. Technical report. EECS Department, University of Calif-ornia, Berkeley, 2016.

[61] B. Hu. Hummingbird. https://github.com/SI-RISCV/e200_ open-source/blob/master/doc.

[62] VexRiscv. https://github.com/SpinalHDL/VexRiscv.

[63] Voung J W, Jhala R, Lerner S. RELAY: Static Race Detection on Millions of Lines of Code[C]. *The 6th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering - ESEC-FSE '07*, 2007: 205-214.

[64] CVE-2010-3190: Untrusted search path vulnerability in the Micr-osoft foundationclass(mfc) library. https://nvd.nist.gov/vuln/ detail/CVE-2010-3190.

[65] D. Svoboda And Y. Toda, Anatomy of Another Java Zero-Day Exploit. https://oracleus.activeevents.com/2014/connect/sessionDetai-l.ww?SESSION ID=2120. Sept. 2014.

[66] CVE-2017-11176: The mq notify function in the linux kernel al-lows attackers to cause a denial of service or possibly have uns-pecified other impact. https://nvd.nist.gov/vuln/detail/CVE-2017-11176. 2017.

[67] CVE-2018-8822: Incorrect buffer length handling in the ncp_rea-d_kernel function could be exploited by malicious ncpfs servers to crash the kernel or execute code. https://nvd.nist.gov/ vuln/detail/CVE-2018-8822. 2017.

[68] CVE-2009-2493: Active template library does not properly restr-ict use of ole load from stream in instantiating objects from data streams,which allows remote attackers to execute arbitrary code. https://nvd.nist.gov/vuln/detail/CVE-2009-2493. 2009.

[69] CVE-2017-12629:Remote code execution occurs in apache solr. https://nvd.nist.gov/vuln/detail/CVE-2017-1262. 2017.

[70] CVE-2015-4422: in huawei mate7. https://cve.mitre.org/cgi-bin/ cvename.cgi?name=CVE-2015-4422. 2015.

[71] Jana S, Narayanan A, Shmatikov V. A Scanner Darkly: Protecting User Privacy from Perceptual Applications[C]. *2013 IEEE Symposium on Security and Privacy*, 2013: 349-363.

[72] Chen X, Garfinkel T, Lewis E C, et al. Overshadow: a virtualization-based approach to retrofitting protection in commodity op-erating systems[C]. *The 13th International Conference on Archit-ectural Support for Programming Languages and Operating Sys-tems*, 2008: 2-13.

[73] Hofmann O S, Kim S, Dunn A M, et al. InkTag: Secure Applications on an Untrusted Operating System[J]. *ASPLoS Proceedings International Conference on Architectural Support for Programming Languages and Operating Systems*, 2013: 253-264.

[74] Criswell J, Dautenhahn N, Adve V. Virtual Ghost[J]. *ACM SIGARCH Computer Architecture News*, 2014, 42(1): 81-96.

[75] McCune J M, Parno B J, Perrig A, et al. Flicker[J]. *ACM SIGOPS Operating Systems Review*, 2008, 42(4): 315-328.

[76] Bickford J, O'Hare R, Baliga A, et al. Rootkits on Smart Phones: Attacks, Implications and Opportunities[C]. *The Eleventh Workshop on Mobile Computing Systems & Applications - HotMobile '10*, 2010: 49-54.

[77] Sailer R, Zhang X, Jaeger T, et al. Design and Implementation of a TCG-based Integrity Measurement Architecture[C]. *The 14th USENIX Security symposium*, 2004: 223-238.

[78] McCune J M, Li Y L, Qu N, et al. TrustVisor: Efficient TCB Reduction and Attestation[C]. *2010 IEEE Symposium on Security and Privacy*, 2010: 143-158.

[79] Havet A, Pires R, Felber P, et al. SecureStreams: A Reactive Middleware Framework for Secure Data Stream Processing[C]. *DEBS '17: The 11th ACM International Conference on Distributed and Event-based Systems*. 2017: 124-133.

[80] Ierusalimschy R. Programming in lua[M]. Roberto Ierusalimschy,2006.

[81] Merkel D. Docker: Lightweight Linux Containers for Consistent Development and Deployment[EB/OL]. 2014

[82] Arnautov S, Trach B, Gregor F, et al. SCONE: Secure linux co-ntainers with intel SGX[C].*12th USENIX Symposium on Operating Systems Design and Implementation*, 2016: 689-703.

[83] Baumann A, Peinado M, Hunt G. Shielding Applications from an Untrusted Cloud with Haven[J]. *ACM Transactions on Computer Systems*, 2015, 33(3): 1-26.

[84] Schuster F, Costa M, Fournet C, et al. VC3: Trustworthy Data Analytics in the Cloud Using SGX[C]. *2015 IEEE Symposium on Security and Privacy*, 2015: 38-54.

[85] Anati I, Gueron S, Johnson S, et al. Innovative technology for CPU based attestation and sealing[C]. *The 2nd in-ternational wo-rkshop on hardware and architectural support for s-ecurity and privacy*, 2013, 13:7.

[86] McKeen F, Alexandrovich I, Berenzon A, et al. Innovative instr-uctions and software model for isolated execution[J]. *The Second Workshop on Hardware and Architectural Support for Security and Privacy,* 2013, 10(1).

[87] Ta-Min R, Litty L, Lie D. Splitting Interfaces: Making Trust between Applications and Operating Systems Configurable[C]. *OSDI '06: The 7th symposium on Operating systems design and implementation*. 2006: 279-292.

[88] Guan L, Liu P, Xing X Y, et al. TrustShadow: Secure Execution of Unmodified Applications with ARM TrustZone[C]. *The 15th Annual International Conference on Mobile Systems, Applications, and Services*, 2017: 488-501.

[89] Park H, Zhai S, Lu L, et al. StreamBox-TZ: secure stream analytics at the edge with TrustZone[C]. *2019 USENIX Annual Technical Conference,* 2019: 537-554.

[90] Trippel T, Weisse O, Xu W Y, et al. WALNUT: Waging Doubt on the Integrity of MEMS Accelerometers with Acoustic Injection Attacks[C]. *2017 IEEE European Symposium on Security and Privacy*, 2017: 3-18.

[91] Hua Z, Gu J, Xia Y, et al. vTZ: Virtualizing ARM TrustZone[C]. *The 26th USENIX Security Symposium*, 2017: 541-556.

[92] Lipp M, Gruss D, Spreitzer R, et al. Armageddon: Cache attac-ks on mobile devices[C]. *The 25th USENIX Security Symposium*, 2016:549-564.

[93] Becher A, Benenson Z, Dornseif M. Tampering with motes: Real-world physical attacks on wireless sensor networks[C]. The 3th *International Conference on Security in Pervasie Computing*, 2006: 104-118.

[94] Halderman J A, Schoen S D, Heninger N, et al. Lest we Remember[J]. *Communications of the ACM*, 2009, 52(5): 91-98.

[95] Becher M, Dornseif M, Klein C N. FireWire: all your memory are belong to us[J]. *Proceedings of CanSecWest*, 2005:67.

[96] Colp P, Zhang J W, Gleeson J, et al. Protecting Data on Smartphones and Tablets from Memory Attacks[C]. *The Twentieth International Conference on Architectural Support for Programming Languages and Operating Systems*, 2015: 177-189.

[97] Zhang N, Sun K, Lou W J, et al. CaSE: Cache-Assisted Secure Execution on ARM Processors[C]. *2016 IEEE Symposium on Security and Privacy*, 2016: 72-90.

[98] Klein G, Norrish M, Sewell T, et al. seL4: Formal Verification of an OS Kernel[C]. *The ACM SIGOPS 22nd symposium on Operating systems principles - SOSP '09*, 2009: 207-220.

[99] Liedtke J. Toward Real Microkernels[J]. *Communications of the ACM*, 1996, 39(9): 70-77.

[100] McKeen F, Alexandrovich I, Anati I, et al. Intel® Software Guard Extensions (Intel® SGX) Support for Dynamic Memory Management Inside an Enclave[C]. *The Hardware and Architectural Support for Security and Privacy 2016 on - HASP 2016*, 2016: 1-9.

[101] Lee R B, Kwan P C S, McGregor J P, et al. Architecture for Protecting Critical Secrets in Microprocessors[C]. *32nd International*

Symposium on Computer Architecture, 2005: 2-13.

[102] Costan V, Lebedev I, Devadas S. Sanctum: Minimal hardware extensions for strong software isolation[C]. *The 25th USENIX Security Symposium*, 2016: 857-874.

[103] Pulte C, Pichon-Pharabod J, Kang J, et al. Promising-ARM/RISC-V: A Simpler and Faster Operational Concurrency Model[C]. *PLDI 2019: The 40th ACM SIGPLAN Conference on Programming Language Design and Implementation*. 2019: 1-15.

[104] Lee D, Kohlbrenner D, Shinde S, et al. Keystone: A framework for architecting tees[DB/OL]. 2019: arXiv preprint arXiv:1907.10119

[105] Asanovic K, Patterson D A, Celio C. The berkeley out-of-order machine (boom): An industry-competitive, synthesizable, parameterized-risc-v processor[R]. Technical report. University of Califo-rnia at Berkeley United States, 2015.

[106] Weiser S, Werner M, Brasser F, et al. TIMBER-V: Tag-Isolated Memory Bringing Fine-Grained Enclaves to RISC-V[C]. *Network and Distributed System Security Symposium*, 2019: 40-55.

[107] Nasahl P, Schilling R, Werner M, et al. HECTOR-V: A Heterogeneous CPU Architecture for a Secure RISC-V Execution Environment[EB/OL]. 2020: arXiv: 2009.05262[cs.CR]. https://arxiv.org/abs/2009.05262

[108] McGillion B, Dettenborn T, Nyman T, et al. Open-TEE —— an Open Virtual Trusted Execution Environment[C]. *2015 IEEE Trustcom/BigDataSE/ISPA*, 2015: 400-407.

姜超 现于上海大学计算机工程与科学学院攻读硕士研究生。研究领域为物联网边缘安全流处理、场景描述语言设计与渲染。Email: superJiang@shu.edu.cn

李玉峰 2008 年于解放军信息工程大学获得通信与信息系统博士。现任上海大学特聘教授。研究领域为网络信息安全管控、智能网联系统内生安全、网络体系结构等。Email: liyufeng_shu@shu.edu.cn

曹晨红 2018 年于浙江大学计算机科学与技术专业获得博士学位。现任上海大学计算机工程与科学学院讲师。研究领域为网络测量与安全、物联网系统与网络、智能感知技术, 边缘计算等。Email: caoch@shu.edu.cn

李江涛 2018 年于华东师范大学获得博士学位。现任上海大学计算机工程与科学学院讲师。研究领域为公钥密码学、车载网安全与隐私保护等。Email: lijiangtao@shu.edu.cn