

## SGX 在可信计算中的应用分析

赵波<sup>1</sup>, 袁安琪<sup>1</sup>, 安杨<sup>2</sup>

(1. 武汉大学国家网络安全学院, 湖北 武汉 430072;

2. 武汉大学计算机学院, 湖北 武汉 430079)

**摘要:** 可信计算技术 SGX 通过隔离出一块可信执行环境, 来保护关键代码及数据的机密性与完整性, 可以帮助防范各类攻击。首先, 介绍了 SGX 的研究背景和工作原理, 分析了 SGX 在可信计算领域的研究现状。然后, 整理了 SGX 当前的应用难点和解决办法, 并与其他可信计算技术进行了对比。最后, 探讨了 SGX 技术在可信计算领域的发展方向。

**关键词:** 可信计算; SGX; 可信性

**中图分类号:** TP311

**文献标识码:** A

**DOI:** 10.11959/j.issn.2096-109x.2021066

## Application progress of SGX in trusted computing area

ZHAO Bo<sup>1</sup>, YUAN Anqi<sup>1</sup>, AN Yang<sup>2</sup>

1. School of Cyber Science and Engineering, Wuhan University, Wuhan 430072, China

2. School of Computer Science, Wuhan University, Wuhan 430079, China

**Abstract:** The trusted computing technology SGX protects the confidentiality and integrity of key codes and data by isolating a trusted execution environment, which can help prevent all kinds of attacks. Firstly, the research background and working principle of SGX were introduced, the research status of SGX in the field of trusted computing were analyzed. Then, sorted out the current application difficulties and solutions of SGX were sorted out and compared with other trusted computing technologies. Finally, SGX technology development direction in the field of trusted computing was discussed.

**Keywords:** trusted computing, software guard extensions, dependability

收稿日期: 2020-12-23; 修回日期: 2021-01-22

通信作者: 袁安琪, 2019202210002@whu.edu.cn

基金项目: 国家自然科学基金联合基金 (U1936122); 湖北省重点研发计划 (2020BAB101, 2020BAA003)

**Foundation Items:** The Joint Funds of the National Natural Science Foundation of China (U1936122), The Key R&D of Hubei Province (2020BAB101, 2020BAA003)

**论文引用格式:** 赵波, 袁安琪, 安杨. SGX 在可信计算中的应用分析[J]. 网络与信息安全学报, 2021, 7(6): 126-142.

ZHAO B, YUAN A Q, AN Y. Application progress of SGX in trusted computing area[J]. Chinese Journal of Network and Information Security, 2021, 7(6): 126-142.

## 1 引言

随着互联网在人们日常生活中扮演着越来越重要的角色,信息安全问题越来越多,无论是智能手机,还是新兴起的无人驾驶、网络电视等都有着不同程度、不同方面的安全问题。小到个人隐私信息,大到国家军事机密,各个层面产生的安全问题复杂烦琐,没有统一的解决方法,但可信计算的出现使从根本上防范信息安全问题的产生成为可能。

在这样的大环境下,英特尔公司产生了研发 SGX (software guard extensions) 技术的灵感,SGX 作为英特尔开发的可信计算技术,旨在以硬件安全为强制性保障,不依赖于固件和软件的安全状态,通过一组新的指令集扩展与访问控制机制,实现包含敏感数据的程序和普通程序间的隔离运行,为包含敏感数据的程序提供可信执行环境,保障用户关键代码和数据的机密性与完整性不受恶意软件的破坏。与其他可信计算技术不同的是 SGX 的可信计算基 (TCB, trusted computing base) 仅包括硬件,避免了基于软件的 TCB 自身存在软件安全漏洞与威胁的缺陷,极大地提升了对系统安全的保障能力;此外,SGX 可以保障可信执行环境运行时的安全,恶意代码无法访问与篡改其他程序在安全区内部运行的内容,进一步增强了系统的安全性;基于指令集的扩展与独立的认证方式,使应用程序可以灵活地调用这一安全功能并进行验证<sup>[1]</sup>。

SGX 的独特架构让它能够在计算平台上提供一个可靠的空间,保障用户关键代码和数据的机密性、完整性和安全区本身,SGX 作为英特尔公司为可信计算推出的一种技术,在被提出后吸引了学术界和工业界大量研究者的关注,并被应用在很多领域。SGX 被大量应用在云计算安全领域,如构建云平台应用安全隔离执行环境、构建安全容器、构建云平台大数据安全可信计算环境和网络功能虚拟化的状态保护。例如, Baumann 等<sup>[2]</sup>提出 Haven, 它利用 Intel SGX 的硬件保护来防御特权代码和物理攻击,是第一个在商用 OS 和商用硬件上屏蔽执行未修改的遗留应用程序的系统。Arnautov 等<sup>[3]</sup>

设计了 SCONE, 一个用于 Docker 的安全容器环境,使用 SGX 技术来保护 Docker 容器内的进程免受外部攻击。Schuster 等<sup>[4]</sup>提出基于 SGX 的 VC3 框架,在保持代码和数据机密性的同时允许用户在云中运行分布式 MapReduce 计算,并确保结果的正确性和完整性。Shih 等<sup>[5]</sup>提出了 S-NFV 方案,结合了 SGX 对网络功能虚拟化产生的状态进行安全隔离保护,来达到安全隔离 NFV 应用程序状态的目的。

除了应用于云计算外,许多解决方案受益于 SGX 提供的安全保护,包括多方计算、机器学习、区块链、人工智能、生物识别技术保护等。例如, UCloud<sup>[6]</sup>引入了与区块链相连接的英特尔 SGX 技术,为智能合约提供了区块链所不能提供的机密性,可拓展的计算节点解决了区块链本身无法应对复杂应用场景的问题,通过可信硬件执行机密性高、需要相对复杂计算能力的程序,并通过区块链对执行结果进行记录和验证,实现可追溯的特性。

SGX 技术目前被广泛地应用到各个领域,但作为可信计算技术而诞生的 SGX 技术现阶段在可信计算领域的应用程度并没有达到相对应的高度。本文针对这一现象,对现有的 SGX 在可信计算领域的研究和应用进展进行了分析,总结出当前阶段 SGX 在可信计算领域难以普及应用的原因及应对方法,并对 SGX 在可信计算领域未来的发展趋势进行了展望。

## 2 SGX 概述

计算机系统处理着快速增长的重要、敏感和有价值的信息,这些信息需要被保护起来,从而避免被篡改和窃取。大量的应用程序需要在平台上保密运行,从而保证用户的敏感数据只能由指定接收人访问。因此, Intel SGX 作为一种新加入英特尔架构的指令集和内存访问技术应运而生。通过应用这项技术可以有效减少攻击面。有无 Intel SGX 安全区 (enclave), 攻击面的不同情况如图 1 所示<sup>[7]</sup>。

### 2.1 SGX 保护机制

#### 2.1.1 SGX 设计原理

英特尔软件防护扩展 SGX 是一项面向应用

程序开发人员的英特尔技术，其目标是对机密信息启用高级别的保护。英特尔从第 6 代英特尔酷睿处理器平台开始引入了英特尔软件防护扩展新指令集，受保护的内存区域称为安全区，使用特殊指令和软件可将应用程序代码放入一个安全区中执行。它通过在代码和数据驻留的地方分配硬件保护的内存来实现安全区和非安全区的隔离。安全区内存中的数据只能由同样驻留在安全区内空间中的代码访问。从安全区的角度来看，操作系统和 VMM 也不是可信的组件。安全区代码可以通过特殊的指令调用。安全区可以构建并加载为 DLL 文件。通过这种机制，安全区可以提供隔离的可信执行环境，防止恶意软件影响安全区内的代码和数据，从而保障用户的关键代码和数据的机密性和完整性。在安全区中运行的受保护程序还拥有密码学测度，用来发送给客户端验证程序的可信执行，或者为远程终端或不可信平台提供认证服务，来实现安全区间的可信性校验。

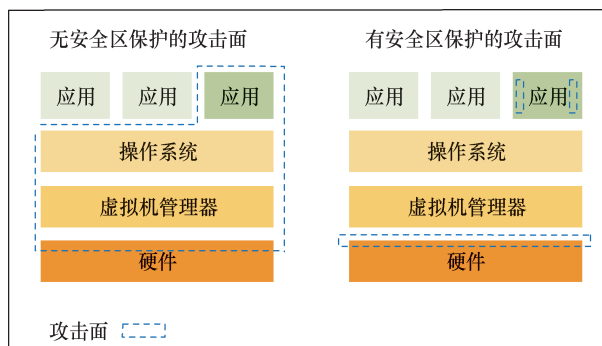


图 1 有无英特尔 SGX 安全区时攻击面的差异  
Figure 1 Differences in attack surfaces with and without Intel SGX enclaves

这种隔离由引导期间留出的处理器保留存储器强制执行。该存储器仅可由 SGX 微码和在安全区内运行的程序访问，并且被分成 4 kB 的簇，这些页统称为安全区页面缓存（EPC，enclave page cache）。EPC 中的页面与特定的安全区相关联，并且只能由该安全区访问。从 EPC 分页到常规 DRAM 的信息在硬件派生密钥下加密<sup>[7]</sup>。

安全区具有多个属性：

1) 通过 CPU 强制执行的内存访问控制机制，将安全区与其他不受信任的应用程序（包括更高权限的应用程序）隔离开；

2) 内存加密用于防御物理攻击并保护交换的安全区页面；

3) 安全区支持安全区实例层面的远程认证。

### 2.1.2 SGX 关键技术

#### (1) 认证

在 Intel SGX 架构中，认证是指证明在平台上建立特定安全区的过程，向其他实体演示以正确的方式实例化特定的环境。SGX 的认证功能目前有两种认证机制。

本地认证：本地认证是指在同一平台上的两个安全区在交换信息之前使用 Intel SGX 报告机制彼此进行认证。在 SGX 应用程序中，当应用程序具有多个需要一起工作以完成任务的安全区，或者当两个单独的应用程序必须在安全区之间传递数据时，可以使用本地认证功能。每个安全区必须验证另一个安全区以确认它们都是可信的。在两个安全区验证对方是可信的之后，可以在受保护的通道上交换信息，该通道可以提供机密性、完整性和重放保护。完成后，两个安全区会建立受保护的会话并使用 ECDH 密钥交换来共享会话密钥。该会话密钥可用于加密必须在两个安全区之间共享的数据。

远程认证：远程认证是指将 Intel SGX 软件和平台硬件的信息组合起来用于生成评价，然后将该评价发送到第三方服务器以建立信任。该软件包括应用程序的安全区，以及由 Intel 提供的 QE（quoting enclave）和 PvE（provisioning enclave）。在证明硬件是支持 Intel SGX 的 CPU 后，软件信息的摘要会与来自硬件平台的唯一非对称密钥组合以生成评价，该评价通过经认证的信道发送到远程服务器。如果远程服务器确定该安全区已正确实例化并且正在支持正版 Intel SGX 的处理器上运行，则它可以信任该安全区并选择使用经过身份验证的通道向其提供机密信息。

#### (2) 密封数据

一个安全区被实例化之后，硬件层面会对数据提供保护，但当安全区被关闭时，安全区内部的所有数据都将丢失，如果数据之后仍然需要使用，那么安全区必须要对这部分数据进行特殊安排，使它们能够存储在安全区外。为了保护 and 保存数据，SGX 建立了一种机制，允许安全区软件检索该安全区的唯一密钥。此密钥只能由特定平

台上的安全区生成。安全区软件使用该密钥加密平台上的数据或解密平台上已经存在的数据。英特尔将这些加密和解密操作分别称为密封和解封操作,因为数据是通过加密方式密封到安全区和平台的。密封数据就是对数据进行加密的过程,以便将其写入不受信任的内存或存储,而不泄露数据内容,且对应的安全区可以将数据读出并解密。加密密钥是按需内部派生的,不会暴露在安全区中。当前 SGX 有两种密封数据的方法。

**密封到安全区标识:**此方法生成某一安全区的唯一指定密钥。当密封到安全区标识时,密钥对于密封数据的特定安全区是唯一的,并且影响其签名的安全区的任何改变都将导致产生新密钥。使用这种方法,由一个版本的安全区密封的数据是其他版本的安全区无法访问的,所以这种方法的副作用是密封数据无法迁移到应用程序及其安全区的较新版本。它是专为密封的旧数据不应被新版应用使用的应用而设计的。

**密封到密封标识:**此方法生成一个基于安全区密封授权方标识的密钥。来自同一签名授权方的多个安全区可以导出相同的密钥。当密封到密封标识时,来自同一授权方的多个安全区可以透明地密封和解封彼此的数据。这允许来自一个版本的安全区的数据被迁移到另一个版本,或者在自同一软件供应商的多个应用程序之间共享安全区中的数据。

如果需要阻止较旧版本的软件和安全区访问由较新的应用程序版本密封的数据,则授权机构可以在签署安全区时选择包含软件版本号(SVN)。早于指定 SVN 的安全区版本将无法派生密封密钥,因此无法对数据进行解封<sup>[7]</sup>。

## 2.2 SGX 管理机制

### 2.2.1 SGX 内存管理机制

SGX 定义了一种全新的数据结构用于维持需要用 SGX 进行保护的关键信息。Enclave 关键数据结构和组件如表 1 所示。

为了能完整地支持 SGX 的内存保护功能,相应的数据结构和硬件结构是必要的。为此,英特尔设计了 EPC, EPC 是 PRM 中操作系统分配的装载应用程序数据和代码的 4 kB 大小内存

的集合,是一个内存存储安全区的页和 SGX 数据结构的受保护内存,这段内存可以同时阻止硬件和软件的访问。处理器会维持整个架构的安全性并对数据进行访问信息控制。如果安全区执行内存访问 EPC 时,处理器决定是否允许访问。

表 1 Enclave 关键数据结构和组件  
Table 1 Enclave data structures and components

| 结构                 | 描述                    |
|--------------------|-----------------------|
| Enclave 页面缓存 (EPC) | 在 4 kB 页面中包含受保护的代码和数据 |
| 页面缓存地图 (EPCM)      | 包含 enclave 页的元数据      |
| SGX enclave 控制仓储   | 每一个安全区的元数据            |
| 线程控制结构 (TCS)       | 每一个线程的元数据             |
| VA 页               | 被驱逐的版本阵列              |
| SIGSTRUCT          | Enclave 的前面结构和密封身份    |

在硬件结构中, EPC 的一个页在硬件结构上称作 EPCM (enclave page cache map)。EPCM 维护 EPC 的入口地址,并且包含 CPU 跟踪 EPC 内存页元数据的状态表。它保证每个 EPC 由一个安全区独享<sup>[7]</sup>。

### 2.2.2 SGX 线程管理机制

任何能够将 EPC 页映射到其虚拟地址空间的进程都能够执行安全区代码。当 CPU 执行安全区代码时称为 enclave 模式,此时 CPU 可以访问驻留在 PRM 中的 EPC 页面。线程数由安全区作者决定,然后决定有多少线程可以并发地执行安全区代码。

初始化后的安全区使用 EENTER 指令来跳转进入安全区执行受保护的程序。只有在 RING3 的程序才能够使用 EENTER 指令, EENTER 并不执行特权切换, CPU 依旧在 RING3 的 enclave 模式下执行。EENTER 指令用于在程序控制下进入安全区,需要保存调用者的执行上下文以便在安全区代码返回时恢复跳转前的状态。

EENTER 需要 TCS 的虚拟地址作为输入,接着这条指令将 RIP (EIP) 寄存器修改为 TCS 中保存的入口偏移地址。这个入口点是由安全区应用开发者定义的,任何对此地址的修改都会导致在初始化安全区时产生不一致的密码学测度。这样保证了安全区中的代码只会在被定义好的地址

上通过 ECALL 来调用。实际上,入口点只是安全区内存范围内的函数,安全区作者已显式地将该内存范围定义为安全区代码的入口点。

### 2.2.3 Intel SGX SDK

Intel(R) Software Guard Extensions SDK 提供了以下工具。

1) Edger8r 工具:用于在不受信任的组件和安全区之间生成接口。它通过读取用户提供的安全区描述语言(EDL)文件来生成边缘例程。这些边缘例程定义不受限于信任的应用程序和安全区之间的接口。通常,该工具作为安全区构建过程的一部分自动运行。但是,高级安全区编写器可以手动调用 Edger8r。

2) Enclave 签名工具: Intel(R)软件保护扩展 SDK 提供了一个名为 `sgx_sign` 的工具,用于对安全区进行签名。通常,对安全区进行签名的过程涉及生成包含安全区属性(如安全区度量)的签名结构,一旦在这种结构中对安全区签名,就可以检测到对安全区文件(如代码、数据、签名等)的修改。签名工具还会评估安全区映像的潜在错误,并警告用户潜在的安全危险。`sgx_sign` 通常是由 Intel(R) SGX SDK 中包含的一种配置工具设置的,并在构建过程结束时自动运行。在加载过程中,将检查签名以确认安全区没有被篡改,并且已正确加载。此外,签名工具可用于报告已签名的安全区的元数据信息,并生成对安全区签名者白名单所需的 SIGStruct 文件。

3) Enclave 调试器:帮助调试安全区。只有调试模式的安全区可以用 Intel(R) SGX 调试器进行调试。

4) 性能测量:使用 Intel(R) VTune 放大器有助于测量安全区代码的性能。Intel(R) VTune 放大器应用程序支持一种新的分析类型,称为 SGX 热点,可用于配置英特尔 SGX 安全区应用程序。可以使用 SGX 热点的默认设置来分析应用程序和安全区代码。精确的基于事件的采样(PEBS)有助于配置 Intel SGX 安全区代码。

5) Enclave 内存测量工具:帮助测量安全区在运行时对受保护内存的使用情况。安全区是一个孤立的环境。Intel(R) Software Guard Extensions

SDK 提供了 `sgx_emmt` 工具,用于测量安全区在运行时对受保护内存的实际使用情况。目前,安全区内存测量工具提供以下两个功能:获取安全区的堆栈峰值使用值和获取安全区的堆峰值使用值。

6) CPUSVN 配置工具:有助于在不修改硬件的情况下模拟 CPUSVN 升级/降级场景。CPUSVN 表示 CPU 的安全版本号,它影响密钥派生和报表生成过程。CPUSVN 不是一个数字概念,会随着硬件升级/降级而升级/降级。为了在不修改硬件的情况下模拟 CPUSVN 升级/降级,Intel(R) Software Guard Extensions SDK 提供了一个 CPUSVN 配置工具,用于配置 CPUSVN<sup>[7]</sup>。

## 3 SGX 在可信计算领域的发展现状

SGX 技术的问世,在学术界和工业界引发了研究的浪潮,并在近年来逐渐成熟,SGX 能够构建可信计算环境的特性在各个方向得到了广泛应用和结合。SGX 技术目前在可信计算领域的应用主要分为 3 种,分别是利用 SGX 增强系统的安全性、利用 SGX 增强应用程序的安全性和利用 SGX 增强开发语言的安全性。

### 3.1 SGX 增强系统的安全性

#### (1) SGX 与 TPM 结合

由于 SGX 技术尚具有局限性和不足之处,可以通过将 SGX 和其他技术结合起来以提高系统整体的安全性,达到设计目标。为了提高 SGX 的安全性,Dhiman 等<sup>[8]</sup>提出了 TGX,如图 2 所示,通过将 SGX 和 TPM 结合,帮助 SGX 安全地将加密操作和安全存储外包给 TPM,TPM 提供了一个具有更强隔离性的专用芯片。通过 EAP 和 SGX 对某些 TPM 位置的唯一访问,以及 SGX 创建的度量等功能协同工作既可以增强对 SGX 的关键加密操作的隔离,又可以使 TPM 管理的凭证得到进一步的保护,这种结构还可以规避针对 SGX 的微架构攻击。TGX 使 SGX 能够将其签名和验证机制从处理器移动到 TPM,从而使安全敏感信息在 TPM 之外永远不可用。通过 SGX 和 TPM 的结合构造了一个安全性更强的可信执行环境。

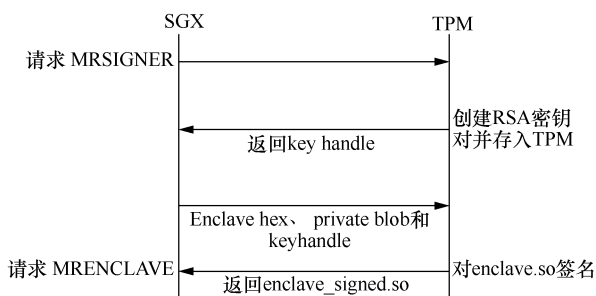


图2 基于TPM在SGX中建立安全区  
Figure 2 Secure enclave build in SGX using TPM

## (2) SGX 与 vTPM 结合

随着云计算技术的广泛应用,SGX 技术已经成为提高云基础设施虚拟化环境安全性的重要手段。根据 TPM 规范,一种 vTPM 的实现框架被提出,为虚拟化环境 vTPM 的设计和实现奠定了基础,但该框架对实例的非易失存储文件 NVRAM 缺少了有效保护;攻击者可以直接读取 NVRAM 文件获取 vTPM 实例中的敏感数据,此外 vTPM 与虚拟机之间不存在绑定关系,攻击者甚至可以直接读取其他虚拟机的 vTPM 实例数据。针对这一安全问题,严飞等<sup>[9]</sup>依据原有架构,将 SGX 的安全特性与 vTPM 结合,提出了 vTSE,其架构如图 3 所示。硬件层面上,使用了支持 Intel SGX 扩展的处理器;SGX 安全区实例中的代码和数据及 SGX 相关的数据结构都存储在 EPC 中,其范围和大小在系统启动时由 BIOS 进行设置。内核层上,通过英特尔提供的 SGX 驱动程序,为应用层提供创建安全区,实现添加/移除安全区页面、销毁安全区等功能。应用层上,虚拟机硬件由 Qemu 模拟,通过调用运行在 SGX 安全区中的 vTPM 库为客户机虚拟出 TPM 设备,由 SGX 为 vTPM 库提供动态保护。

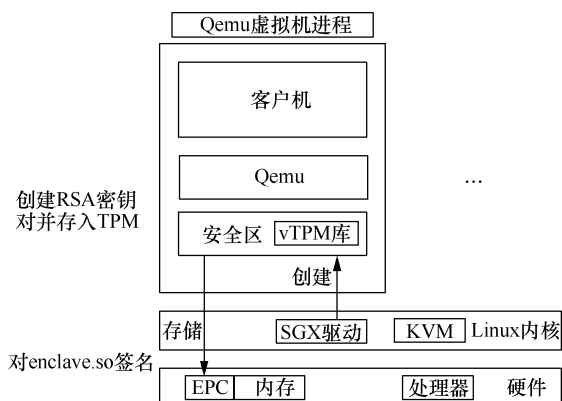


图3 vTSE 架构  
Figure 3 vTSE architecture

## (3) SGX 与 Hybrid Memory Safety 内存安全技术结合

当前对于数据安全和隐私保护的合规性要求日渐提升,如何对用户隐私进行保护、对数据安全可控已经成为广泛的需求,MesaTEE 正是应运而生的一个创新性的通用计算平台。百度与英特尔联合,将 SGX 以及其他可信计算技术和混合内存安全技术结合在一起,构建了完整的 FaaS 通用计算框架,形成了一个可信计算平台 MesaTEE,通过利用包括 Intel SGX 在内的 TEE,安全地将云用户的数据和代码执行隔离、加密。即使云计算环境中的操作系统、虚拟机管理器或相邻的其他虚拟机被攻破,这些数据代码的完整性和保密性能够得到保障,用户也可以远程对这些安全保护进行验证。除了远程可验证的隔离手段,MesaTEE 还实现了全球首次在 SGX TEE 中使用内存安全语言和不可绕过编程范式,确保 SGX 安全区中的代码不会因为内存问题被攻破,从内外两个维度封杀了攻击者突防的可能性<sup>[10]</sup>。

## 3.2 SGX 增强应用程序的安全性

Intel SGX 在应用程序中的部署主要分为本地、服务端和客户端。对于独立运行在本地的程序,SGX 无须与远程进行交互。目前主要用 Intel SGX 存储服务器的敏感数据。在客户端的应用较少,主要通过 SGX 对客户端软件的完整性和安全性进行校验。同时部署在客户端和服务端端的应用可以通过 SGX 实现远程认证。

### (1) SGX 应用于本地

SGX 本身作为一种可信计算技术,能够很好地与系统功能融合,保障系统敏感数据的存储安全。例如,系统日志作为捕获操作系统或程序行为的最大法证资产,一旦系统受到威胁,系统日志通常是下一个立即受到攻击的目标,因此保护它们至关重要。基于这一出发点,Vishal 等<sup>[11]</sup>将 SGX 与日志系统结合,设计出 SGX-Log,这是一种新的日志记录系统 SGX-Log,可确保日志数据的完整性和机密性,其关键思想是利用 SGX 提供的具有密封和启封原语的安全区域,保护内存和磁盘中的程序代码和数据不被未经授权修改。SGX-Log 使用客户端和服务端架构。客户机是一个日志请求组件,它发出各种日志消息,而日志

服务器执行安全日志服务。SGX-Log 提供了启用 SGX 的日志服务器的详细设计、实现和评估。由于 SGX 安全区可用的内存有限,以及与不可信代码交互的成本,必须设计一个日志系统来适应这种情况。SGX-Log 为如何有效地利用 SGX 对日志进行保护提供了具体的结果和见解。

### (2) SGX 应用于服务器端

SGX 具有最高安全权限的特性使 SGX 在保证敏感数据安全性、完整性方面具有巨大的优势,而这种优势可以用来为服务器中用户信息和证书之类的敏感信息提供安全存储的服务。

相对于传统支付而言,区块链具有无须受信任的授权即可进行操作的能力以及保障用户隐私的功能。但是,区块链支付的验证要求用户下载并处理整个链,这对于资源受限的设备而言是不可行的。为了解决这个问题,大多数的区块链系统支持轻量级客户端,这些客户端将大部分计算和存储负担外包给了完整的区块链节点。但是,这样的验证会泄露有关客户交易的关键信息,从而破坏用户隐私。Sinisa 等<sup>[12]</sup>利用 SGX 安全区的可信执行功能提出了一种保护比特币中轻量客户端隐私的新方法,来保护比特币中轻量客户端隐私,他们设计并实现了一个称为 BITE 的系统,该系统中的完整节点上的安全区可满足轻客户端的隐私保护请求。核心思想是在完整的节点中利用 SGX 安全区的可信计算能力,让一个不受信任的实体运行带有 SGX 安全区的完整节点,并且 SGX 安全区对来自客户端的交易确认提供请求服务,通过 SGX 技术提供代码完整性和数据机密性。图 4 展示了由完整节点组成的系统模型。当

轻量级客户端想要获取关于它的事务或地址时,可以连接到任何支持服务的完整节点  $FN_j$ ,并承载一个 enclave  $E_j$ 。完整节点下载并本地存储整个区块链 (BC),基于它维护一个数据库,其中包含所有未使用的事务输出 (UTXO)。

除了敏感信息之外,服务器端的数据库也存在安全隐患,通过将数据库纳入 SGX 的保护下可以有效避免对数据库的 RING0 攻击。为此,Priebe 等<sup>[13]</sup>研发了 EnclaveDB, EnclaveDB 通过将敏感数据(表、索引和其他元数据)放置在受信任硬件(如 Intel SGX)保护的安全区中来实现即使数据库管理员是恶意的,攻击者入侵了操作系统或虚拟机管理程序,或数据库在不受信任的主机中运行,EnclaveDB 仍可以保证数据库安全性的目的。EnclaveDB 具有类似于传统关系数据库的编程模型,经过授权的用户可以创建表和索引,并使用 SQL 表示的存储过程查询表。与传统数据库不同,EnclaveDB 提供了抵御黑客、恶意服务器管理员和数据库管理员攻击的安全性。如图 5 所示,EnclaveDB 通过将所有敏感数据(表、索引、查询和其他中间状态)托管在 enclave 内存中来保护数据库状态。这种设计选择是可行的,因为 DRAM 成本迅速下降,而且支持大型集群的系统的预期可用性很高。

### (3) SGX 应用于客户端

SGX 在客户端的应用虽然较少,但是可以满足软件在客户端的完整性和安全性认证,可用于开发商向用户客户端的 Intel SGX 中下放许可证等版权信息。SGX 客户端应用包括 SGX 的分布式客户端应用。

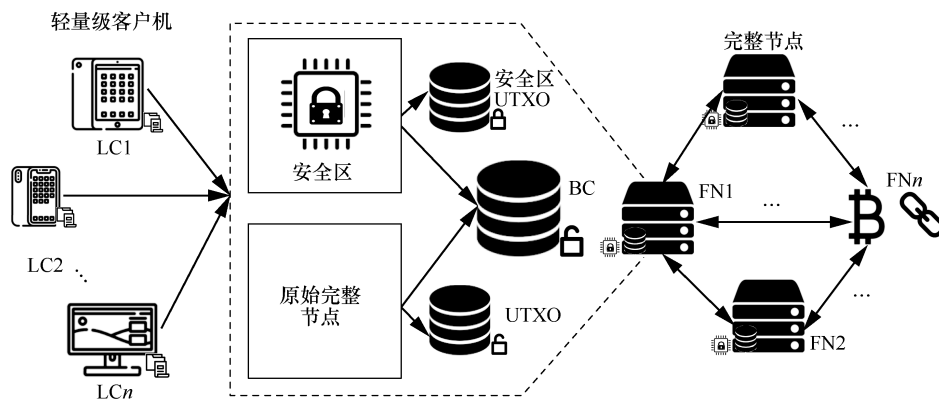


图 4 轻量级客户端向安全区请求事务验证的工作流程

Figure 4 Workflow for lightweight clients to request transaction validation from enclave

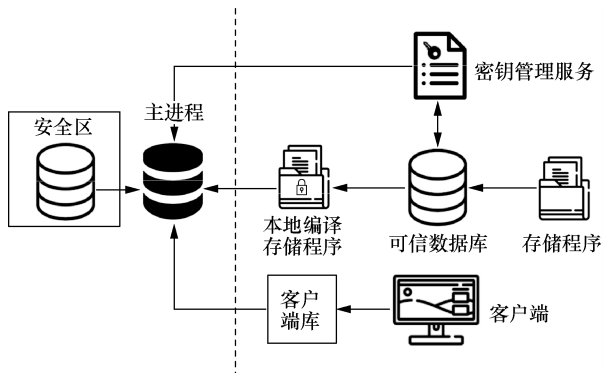


图 5 EnclaveDB 的体系结构  
Figure 5 EnclaveDB's architecture

长期以来, 计算机游戏的完整性和保密性一直是游戏开发商关注的问题, 为了防止玩家作弊或获得未经授权的软件复制, 游戏厂商在反向上付出了大量的精力, Bauman 等<sup>[14]</sup>利用 SGX 基于硬件的保证且可以维护软件运行态安全的功能来保护游戏的机密性和完整性, 而且可能被广泛部署在用户机器上的特点, 提出了基于 SGX 保护游戏完整性和机密性的一般框架, 该原型框架可以集成到现有的游戏中。

对于游戏软件完整性的保证, 由于不同的游戏有不同种类的信息, 为了防止作弊, 必须保护这些信息。为了完整性而必须保护的代码或数据取决于被保护的特定游戏, 找到目标游戏的这些敏感代码或数据后, 必须将其转移到安全区内。当用户启动应用程序时, 安全区将与受信任的服务器联系并执行远程认证。通过验证的安全区则认为其中的代码和数据没有被修改。保证敏感数据和代码完整性的机制具体如图 6 所示。

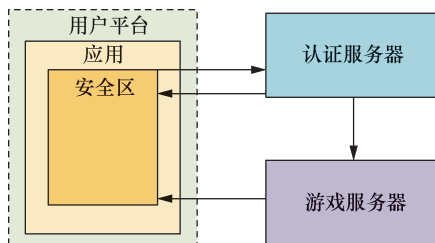


图 6 保证敏感数据和代码完整性的机制  
Figure 6 Mechanisms to ensure the integrity of sensitive data and code

对于游戏软件机密性的保证, 如何处理游戏资产有两种选择。第一个选项是将解密的代码或数据直接保存在安全区中, 这对于少量关键代码或数据非常适用。如果数据是加密的图像, 那么

一次性将其全部保存在安全区中是不可行的。因此, 第二个选项是立即将解密的数据返回到主内存中的主应用程序。在这种情况下, 需要被隐藏的敏感数据并不是图像, 而是解密密钥和加载机制。对于通常存储在磁盘上的大型资产, 安全区可以简单地密封解密密钥。这通常不需要任何外部连接的单人游戏只需要一个连接来验证服务器并获取解密密钥。之后, 就可以自动操作, 而无须再次与服务器签入。保证敏感数据和代码机密性的机制如图 7 所示。

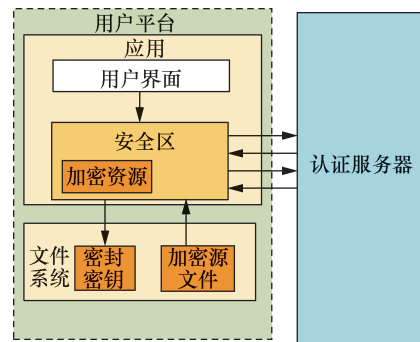


图 7 保证敏感数据和代码机密性的机制  
Figure 7 Mechanisms to ensure the confidentiality of sensitive data and code

客户端 JavaScript 在 Web 应用程序中无处不在, 用于改善用户体验和减少服务器负载。由于客户端不受信任, 服务器不能依赖客户端 JavaScript 代码和它操作的数据的机密性或完整性, 如必须在服务器端重复客户端输入验证, 并且不能卸载机密业务逻辑。所以, Goltzsche 等<sup>[15]</sup>提出了 TRUSTJS 框架, 利用由 Intel SGX 提供的可信硬件支持, 以保护 JavaScript 的客户端执行, 实现了 Web 应用程序代码的灵活分区, 使 JavaScript 可以在商品浏览器中可靠地执行。

#### (4) SGX 应用于服务器端和客户端

当 SGX 同时部署在服务器端和客户端时, SGX 的远程认证功能可以在同时具有 SGX 功能的服务器端和客户端之间实现安全通信, 冯达等<sup>[16]</sup>利用 SGX 可以存储敏感数据并进行远程认证的功能, 实现了服务器端和客户端之间证书的安全交互, 将 SGX 技术与数字签名、加密技术相融合, 实现了证书可信性验证与软件安全签发系统。

在多方计算领域, SGX 的出现提供了新的安全思路。Olga 等<sup>[17]</sup>提出了一种基于可信 SGX 处

理器的保密性多方机器学习系统。在该系统中,多方在一个联合的机器学习任务上执行它们的聚合数据,并在一个启用 SGX 的数据中心上运行该任务。每一方都独立地与安全区建立一个安全通道,用于对自己进行身份验证,并可以独立地与云提供商交互,使用远程认证的功能来确认 SGX 处理器是该数据中心的一部分,检查加载到安全区的代码和静态数据的完整性。每一方都使用一个单独的、本地生成的密钥来加密自己的输入数据集,并使用其安全通道与安全区共享该密钥。如图 8 所示,基于这个系统可以实现多家医院用不同的密钥加密患者数据集。医院在云数据中心的安全区中部署一种商定的机器学习算法,并与安全区共享它们的数据密钥。安全区处理聚合数据集并输出加密后的机器学习模型。

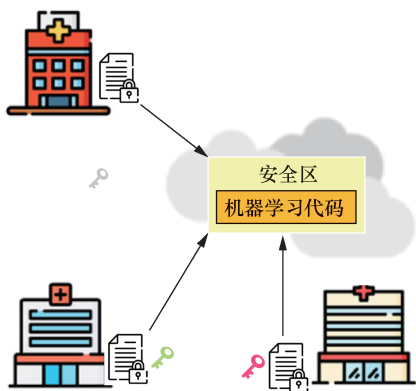


图 8 多家医院用不同的密钥加密患者数据集  
Figure 8 Multiple hospitals encrypt patient data sets with different keys

### 3.3 SGX 增强开发语言的安全性

虽然研究人员提出了许多基于 SGX 提高应用程序安全性的解决方案,但思路大多是保护本机代码应用程序,而脚本语言并不受保护。为了填补这一空白,Wang 等<sup>[18]</sup>提出了 SCRIPTSHIELD,这是一个能够向后兼容的应用程序执行框架,通过在 SGX enclaves 内执行脚本语言解释器,自动确保脚本的机密性和完整性,核心思想是自定义脚本解释器,使其在 SGX 安全区中运行并将脚本传递给它。

## 4 SGX 在可信计算领域应用的难点及解决方法

虽然对 SGX 的研究进行了多年,技术已经趋

于成熟,甚至已经投入工业界的实际使用,但现阶段 SGX 在可信计算领域的开发和应用并没有呈现出与它热度相匹配的前景,这主要有两个原因,一个是 SGX 本身的开发和应用难度限制了研究人员对它的应用,另一个是 SGX 与其他可信技术相比在应用中有较多问题需要解决。

### 4.1 SGX 开发和应用难点分析及解决方法

SGX 技术自推出以来受到了学术界和工业界的极大关注,目前在很多领域得到了广泛的应用,SGX 技术在研究中暴露出了很多缺陷和问题,如各类针对 SGX 的侧信道攻击表明 SGX 技术当前还存在很多安全隐患。事实上,除了安全性问题,SGX 在研究过程中还暴露出了许多性能瓶颈、开发和部署困难、功能局限等诸多应用方面的问题。当前以 SGX 作为可信基的系统设计及应用主要面临性能、安全性、开发难度和应用 3 个方向的挑战。

#### 4.1.1 性能

SGX 作为可信基的性能开销,主要是从模式切换、Enclave 页替换和内存加密 3 个方面产生的<sup>[19]</sup>。

##### (1) 模式切换

模式切换是 SGX 最主要的性能开销,但也是 SGX 能够在安全区中执行代码的基本保障机制。系统进入或退出安全区都会产生模式切换,模式切换开销的主要原因是出于对安全性的保证,每一次模式切换都要进行复杂的状态保存和安全检查、保存和恢复 CPU 状态,并在安全区和不可信软件进行上下文切换时刷新 CPU 核心的 TLB,由此产生了大量的性能消耗。

当 SGX 作为可信计算基时,对安全区内数据和代码的每一次访问和执行都会导致模式切换的产生,如果系统被设计为需要在安全区和非安全区之间进行大量的数据调用,那么产生的性能开销相对会较高。

降低模式切换开销的技术主要有:无切换调用<sup>[20]</sup>、增加操作系统库和函数库减少模式切换<sup>[21]</sup>和轻量级细粒度的并行减少安全区之间模式转换<sup>[22]</sup>。

##### (2) Enclave 页替换

由于安全区内存空间比较小,SGX 支持从 EPC 到主内存的页替换,以满足不适合 EPC 的安

全区,所以当程序代码数量和规模增大时,敏感页面会在内存中的 EPC 和非 EPC 区域之间频繁交换,从而产生了开销巨大的 Enclave 页替换开销。内存要求超过 EPC 大小的应用程序必须在 EPC 和未受保护的内存之间替换页面。出于对安全性的要求,EPC 页必须在被复制到外部内存之前被加密和完整性保护,工作量较大,代价高昂。为了防止地址转换攻击,需要替换协议中所有 enclave 线程并刷新 TLB。不仅如此,页替换还会导致 enclave 模式的切换,进而产生模式切换开销。

虽然 SGX 的开发原则中始终强调最小特权原则,但是当 SGX 作为可信计算基时,并不仅仅起着对数据进行安全存储的作用,它还能够为安全区内运行的程序提供机密性和完整性的保护。通过在安全区内存储尽量多的数据和运行的代码量,来减少安全区和非安全区之间的交互,虽然可以提升性能,但是增加了攻击面。如何在安全性和性能之间进行平衡是基于 SGX 的系统在设计中亟待解决的重点、难点。

针对页替换开销的主要技术是高效的完整性验证结构、减少分页开销技术和安全用户管理虚拟内存技术,未来,这一性能开销限制可能会从根本上得到改善:虽然目前 SGX v1 版本支持的安全区最大为 128 MB,应用程序实际的使用量一旦超过 93 MB 就会触发频繁的页替换,增加安全风险和性能开销,但在 SGX v2<sup>[23]</sup>中可能可以支持安全区扩展到整个内存。

### (3) 内存加密

SGX 通过内存加密技术保护程序运行时的安全,MEE 必须加密和解密高速缓存线,会导致加密内存访问存在高昂的开销,安全区代码也会由于写入内存和高速缓存未命中产生开销,这就是内存加密开销。

SGX 作为可信计算基除了存储敏感数据的功能外,和其他传统可信计算技术相比,最大的优势就是能够保证安全区内程序运行过程中的安全,而这一重要功能必须靠内存加密技术来保障,无法绕过内存加密产生的性能开销问题。SCONE<sup>[3]</sup>使用 SGX SDK 微基准测试评估了在安全区中进行顺序读/写和随机读/写操作的时间开销,得出“在 EPC 大小范围内几乎没有内存访问

开销,而超出 EPC 大小的内存访问有 2 倍左右性能开销”的结论。Eleos<sup>[24]</sup>通过生成 10 万个随机数证明内存随机访问和连续访问,以及读和写操作的开销相近,介于 5.6 倍到 9.5 倍开销之间。这些实验说明了内存加密开销是 SGX 设计本身导致的,难以进一步优化。

基于 SGX 的应用目前只能通过减少对安全区的不必要使用和使用节省空间的数据结构或将较小的数据块加载到安全区中来缓解开销。安全程序设计应尽量减少对安全区内存的使用和访问来降低内存访问开销。如果必须使用安全区内存,则尽量采用连续访问的数据结构,避免采用随机访问的数据结构。

### 4.1.2 安全性

在安全性方面,SGX 虽然在原则上可以作为可信执行环境,但是近年来大量的工作表明,SGX 自身的机制导致它的安全区很容易被各类攻击攻破,通过攻击获得的敏感数据可以帮助攻击者绕过 SGX 提供的数据机密性保证。一旦可信计算基的某个构件出现程序错误或者安全隐患,会对整个系统的安全造成危害,SGX 的设计目的是为安全区内的敏感数据提供机密性和完整性保证,现阶段它在实际应用中依然会面临诸多的安全漏洞和威胁,这也是 SGX 无法投入实际应用的重要原因之一。

#### (1) SGX 自身安全缺陷

由于 SGX 的设计机制和微架构带来的安全问题,针对 SGX 本身的安全攻击主要分为侧信道攻击和高低电压攻击\高低频率攻击两类。

基于 SGX 常见的侧信道攻击有 6 类,分别是基于页表的攻击、基于 TLB 的攻击、基于 Cache 的攻击、基于 DRAM 的攻击、基于 CPU 内部结构的攻击和混合侧信道攻击,如较为经典的 Meltdown<sup>[25]</sup>、SgxPectre<sup>[26]</sup>、Foreshadow<sup>[27]</sup>和 RIDL<sup>[28]</sup>攻击。

现阶段针对侧信道攻击的解决方法主要有 4 种:第一种是在源码层次修改源码,主要思路是隐藏控制流和数据流、改善加密算法;第二种是在系统层次利用一些系统特性来防御或检测 SGX 侧信道,如随机化技术、检测可疑异常和中断、检测时间异常、对 Cache 隔离;第 3 种是在应用层

面引入安全机制,防范侧信道攻击(如 Oleksii 等<sup>[29]</sup>提出了 Varys 系统),它可以保护在 SGX 安全区中运行的未修改的程序免受缓存计时和页表侧信道攻击,Varys 通过将物理内核严格保留给安全敏感的线程,从而防止攻击者在安全区执行期间访问共享的 CPU 资源;第4种是在硬件层面进行硬件分割或者硬件隔离,但是这种方法会显著增加硬件复杂度,影响功耗和性能。

除了侧信道攻击外,高低电压攻击\高低频率攻击也能通过恢复加密密钥或在以前安全的软件中引入错误来达到破坏 SGX 安全性的目的。现代处理器正以更快的速度运行,随之而来的是热量和功耗的增加。为了平衡速度和性能,许多芯片制造商允许用户在需要时调整频率和电压,但是随之而来的安全问题是用来修改频率和电压的特权软件界面,可能会被利用来破坏系统的安全性。例如,Plundervolt<sup>[30]</sup>攻击会通过执行安全区计算时的电压,破坏处理器上 SGX 的完整性。在安全区计算期间,Plundervolt 会仔细控制处理器的电源电压,从而在处理器封装内引起可预测的故障。因此,即使是 Intel SGX 的内存加密/身份验证技术也无法防范 Plundervolt 攻击。

针对这一攻击,英特尔发布了相关微代码和 BIOS 更新。这些更新为管理员提供了一个新的 BIOS 选项,可以在管理员认为 Plundervolt 构成真正风险的情况下禁用系统上的电压和频率控制界面。

## (2) 基于 SGX 软件的安全缺陷

除了 SGX 自身设计机制带来的攻击以外,基于 SGX 的应用可能由于设计过程中的忽略而存在安全缺陷,如接口安全和反汇编产生的敏感代码安全性问题。

很多基于 SGX 的应用软件及框架设计有众多缺陷。这些软件架构设计上的疏忽使攻击者无须权限,即可绕过 SGX 安全区的严密封锁,获取存储在 SGX 安全区中的核心敏感数据。Coin Attack<sup>[31]</sup>的出现,揭示了 SGX 在接口安全的维度的弱点。

针对这一对接口安全进行攻击的方式,百度安全实验室开发了一个名为 Coin Attack 的检测框架,通过符号执行的方法,可以捕获各种各样的

漏洞,强调了基于 SGX 的应用不能仅依赖于 SGX 的强隔离保证,还需要对软件架构细心设计,确保健壮安全的接口调用方式。开发者如果忽视软件中的接口安全,整个 SGX 安全区的安全保障都会丧失。

虽然 Intel SGX 提供了一个安全的区域,其中的代码和数据对外界是隐藏的,包括特权代码,但在默认情况下,初始化之前的安全区代码可以反汇编,因此二进制代码中不能嵌入任何秘密。对于希望保护代码机密的开发人员来说,这是一个问题。对此,Bauman 等<sup>[32]</sup>开发了 SGXElide,这是一个几乎透明的框架,可实现安全区代码的机密性。SGXElide 通过将程序代码视为数据,并在初始化安全区域后动态还原机密来提供一种机制,可以在开发人员控制的受信任远程方的协助下安全地解密或传递密码。

## 4.1.3 开发和应用难度

在开发和应用难度方面,SGX 作为最新的可信计算技术,虽然在思想和部分机制上借鉴了传统的可信计算技术,但是在开发和应用上要求更为烦琐,较难上手。同时,基于 SGX 技术开发的软件在设计上要求做到性能和安全性之间的平衡,在不影响用户体验的前提下保证敏感数据都被存储在安全区内,这对程序员对程序安全性的划分能力要求较高。对于已经完成的系统,需要对系统代码进行修改才能在 SGX 上使用,可移植性较差。

### (1) 开发环境

对于一些研究人员缺乏 SGX 开发环境的问题,Jain 等<sup>[33]</sup>开发了名为 OpenSGX 的开源平台,该平台在指令级别上模拟 Intel SGX 硬件组件,并提供了全面 TEE 探索所需的新系统软件组件。这个开源平台开发了 Intel SGX 的全功能,指令兼容的仿真器,用于探索软件/硬件设计空间以及开发安全区程序。OpenSGX 为 SGX 开发提供了一个平台,不仅提供仿真,而且提供操作系统组件、安全区程序加载器/打包器、OpenSGX 用户库、调试和性能监控。由于其用户模式仿真,其系统级别的支持在很大程度上受到限制。针对使用 OpenSGX 无法进行需要系统功能或设备仿真实验的问题,Choi 等<sup>[34]</sup>推出优化了 System-

OpenSGX (S-OpenSGX), 该系统利用 QEMU 的系统仿真为研究人员提供全面的系统级支持, 以探索基于 SGX 安全区的编程。

#### (2) 开发语言

对于 SGX 开发语言类型有限的问题, 开发人员开发出了 Rust SGX SDK<sup>[35]</sup>、MesaPy<sup>[36]</sup>、GOTEE<sup>[37]</sup>等安全开发语言。百度安全实验室将 Rust 语言和 SGX 技术结合, 开发出了 Rust SGX SDK, 开发人员使用 Rust 语言可以快速开发出没有内存安全漏洞的 SGX 应用程序, 避免敏感数据被窃取。MesaPy 是百度安全实验室开发的一个内存安全的 Python 实现。通过使用内存安全语言重写外部库和形式化验证保障代码的内存安全等方法全面提升 Python 解释器的安全性, 避免内存问题引发的高危安全漏洞。基于这些安全特性, MesaPy 可以让开发者使用 Python 轻松地进行 SGX 开发。Ghosn 等通过扩展 Go 语言, 允许开发人员依靠编译器自动提取安全代码和数据, 将可信执行环境继承到了语言中, 增强了 SGX 的可用性。

#### (3) 开发难度

对于 SGX 技术缺少针对安全区间交互的适当编程支持, 并且要求将可信执行的确切使用方式硬编码到应用程序中的限制问题, Sartakov 等<sup>[22]</sup>提出了 EActor 框架, 这种为 SGX 量身定做的 Actor 框架能够提供对可信执行的更无缝、更灵活和更高效的使用, 支持低成本的多安全区编程和交互。

#### (4) 可移植性

对于已经编写完成的程序, 开发人员推出了 Haven<sup>[2]</sup>、Panoply<sup>[38]</sup>和 Graphene-SGX<sup>[39]</sup>, 能够让传统 App 不需要修改代码即可在 SGX 中运行。其中较为完善的是蚂蚁集团使用 Rust 语言编写的 Occlum<sup>[40]</sup>, 将库操作系统引入了 SGX, 从而使遗留应用程序可以在安全区域内运行而几乎不需要修改的基础上, 通过将 LibOS 流程实现为 SFI 隔离流程的方式增加安全高效的多任务处理功能。

### 4.2 SGX 与其他可信计算技术相比的优缺点

除了 SGX 自身在性能、安全性和开发及应用方面的问题以外, 和其他的可信基相比, SGX 独

特的机制既带来了优势, 也导致了它在开发和应用上存在局限性。

#### (1) TPM/TCM

在开发和应用方面, TPM 是当前可信计算领域应用最为常见的可信基, 广泛部署在商用计算机上, 不需要对 CPU 进行修改, 仅依赖于一个辅助的抗篡改芯片, 而 SGX 直接集成在 CPU 内部, 性能较 TPM 得到了提升。相比于 SGX, TPM 的可信计算基范围更大, TPM 信任计算机上的所有软件, 而 SGX 仅信任安全区, 可信计算基的范围虽然使 TPM 可度量的范围更大, 但也导致认证签名中包含的度量覆盖了整个 OS 内核和所有内核模块, 因此需要随着软件更新持续维护对应的可接受测量的哈希表, 实际应用较为烦琐复杂。

在安全性方面, 由于 TPM 晶片没有为计算机上的软件提供任何隔离, 所以基于 TPM 的系统容易受到物理访问机器的攻击者的攻击<sup>[41]</sup>, 且 TPM 作为外挂部件只能被动调用, 不能进行主动度量<sup>[42]</sup>。SGX 在安全性方面虽然也有很多攻击问题亟待解决, 但与 TPM 相比, SGX 不仅可以保障敏感数据存储安全, 还可以确保安全区内部软件运行态的安全。

#### (2) Intel TXT

在开发和应用方面, 英特尔的可信执行技术 (TXT) 使用了 TPM 的软件认证模型和辅助抗篡改芯片, 但将安全容器内的软件减少为一个虚拟机, 由 CPU 的硬件虚拟化特性 (VMX<sup>[43]</sup>) 托管, 主要用于配合英特尔新一代的虚拟化技术来保护虚拟化运算环境下的数据。TXT 技术保护的對象可以理解为整个操作系统, 而 SGX 技术保护的對象是应用程序运行时的一部分代码和相应的地址空间。与 Intel TXT 技术相比, 由于 SGX 的可信范围较小, 只支持 RING3 指令, 导致 SGX 无法实现在不受信任的操作系统下访问磁盘的这类操作, 所以 TXT 的适用范围相较于 SGX 而言更广。

在安全性方面, TXT 没有实现 DRAM 加密或 HMAC, 因此容易受到物理 DRAM 攻击, 而且 hypervisor 潜在的软件漏洞可能威胁到整个系统, 运行在系统管理模式下的软件可以绕过 VMX 使用的限制来提供隔离。但与 SGX 相比 TXT 攻击面较少, 且 TXT 能够检测 VMM 的运行状况,

保证虚拟设备的正常运行,而 SGX 对于运行在安全区内部的软件并没有监管能力。

### (3) ARM TrustZone

在开发和应用方面,与开放了 SDK 的 SGX 相比,TrustZone 需要硬件厂商的签名验证才能运行在安全执行环境,这限制了开发人员基于 TrustZone 进行研究和开发。TrustZone 为程序提供了 TEE 和 REE 两种隔离的执行环境,只有一个可信环境,而 SGX 可以同时运行多个可信环境,这限制了 TrustZone 的应用面。虽然在开发上有限制,但是 TrustZone 被设计为在系统加电后优先获得控制权,并且拥有比主机更高的访问和控制权限<sup>[44]</sup>,因此 TrustZone 比 SGX 更贴合可信计算 3.0“主动防御体系”对主机系统进行主动访问和监控以及系统上电时代码可信性的功能要求,在可信计算 3.0 领域具有设计上的优势。

在安全性方面,SGX 与 TrustZone 相比虽然可以多个安全区并发执行,且原则上即使某个安全区中存在恶意程序,也无法访问或篡改其他安全区内被保护的内容,使隔离环境的安全性虽然有所提高,但是依然有缺陷。TrustZone 具有对非安全世界进行监视的系统,这些监视系统拥有高权限,可以动态地检查 Android 系统的内核本身<sup>[45]</sup>,也可以代替 Android 内核来管理非安全世界的内存。而 SGX 缺乏对平台上恶意软件的识别和隔离,这就导致实际应用时无法检测安全区内部运行着的是否为恶意程序,也无法对恶意程序可能窃取其他安全区的敏感数据的行为进行阻止。

### (4) AMD SEV

在开发和应用方面,SEV 的最小可信基比 SGX 更大,包含整个操作系统或者虚拟机,所以在设计上 SEV 比 SGX 更适合用来保护大规模的计算环境,且 SEV 允许每个虚拟机使用自己的密钥选择性地加密内存,这使 SEV 能够为不可信云环境下的数据提供安全保护<sup>[46]</sup>,更适用于云计算的应用场景。

在安全性方面,相比 SGX 通过内存管理单元来实现内存安全,SEV 使用了安全内存加密(SME)来加密 VM 内存,防止物理攻击和特权软件攻击。但 SEV 较大的信任范围导致了较大的攻击面,当 SEV 的应用环境为不可信的虚拟机管

理程序时,恶意的虚拟机监控器可以通过修改相关的关键数据结构绕过甚至直接关闭 SEV 的保护机制。虽然 SME 对内存进行了加密保护,但 SEV 缺乏对主存储器逐页加密的完整性保护机制,由此产生的 SEVered 攻击使攻击者能够访问经由 SEV 加密之后虚拟机上的明文内存数据。并且,SEV 提供的保护不足以覆盖虚拟机的整个生命周期,在设计上同样没有考虑到侧信道攻击,存在较大的安全隐患。

综上所述,虽然 SGX 与其他可信计算技术相比在设计机制上更加灵活,且便于开发者进行应用开发,但是可信根范围的局限性和亟待解决的安全性问题导致 SGX 与其他可信计算技术相比没有突出优势,在很多场景下其他可信计算技术更符合场景要求,以至于 SGX 在可信计算领域并没有呈现出与其热度相匹配的应用量。

## 5 SGX 在可信计算领域的发展方向

自 2013 年英特尔公司推出 SGX 技术以来,SGX 技术在学术界和工业界引发了广泛的研究浪潮,并被陆续应用到了很多领域。目前,虽然 SGX 作为可信计算基,在以云计算为代表的各领域内开发和应用发展迅速,但是在可信计算领域的研究并没有达到符合预期的水平,尚缺乏安全、有效、彻底的应用方案。基于以上研究进展,本文提出了 SGX 在可信计算研究领域的进一步应用需求和研究方向。

### 5.1 SGX 安全性

SGX 本身的安全漏洞是 SGX 在可信计算领域发展中面临的最根本问题,如果无法解决安全问题,SGX 就不能作为可信计算基为上层架构提供安全保证。如何有效解决 SGX 应用面临的侧信道攻击、内存攻击等安全威胁,增强 SGX 应用的安全性依然是该技术应用研究的重要研究方向。由于 SGX 在安全区和非安全区之间共享资源而产生了攻击面的设计机制,当前的 SGX 自身无法抵御侧信道攻击。现阶段,在软件层面打补丁以及升级微码等解决方案无法从根本上解决 SGX 应对侧信道攻击的问题,而通过引入安全机制来防范侧信道攻击又会引发额外的性能开销,故可以通过将 SGX 与外界的输入输出进

行加密、创建检查点等方式提高 SGX 应对攻击的能力。

除了要面对外部诸如电源攻击之类的威胁, SGX 还需要面对安全区内部恶意程序的威胁。SGX 无法对运行在安全区中的软件进行监管, 恶意软件可能会在安全区内执行其恶意代码, 来达到获取敏感数据甚至安全区密钥的目的, 从而能够有效地绕过 SGX 提供的整体安全性措施。针对这一问题, 未来的发展可以参考 TrustZone 的监管模式, 引入监控的思想, 如当监控器检测到可能导致攻击行为的异常指令时, 强制开启 AEX, 中止安全区的运行, 思路如图 9 所示。

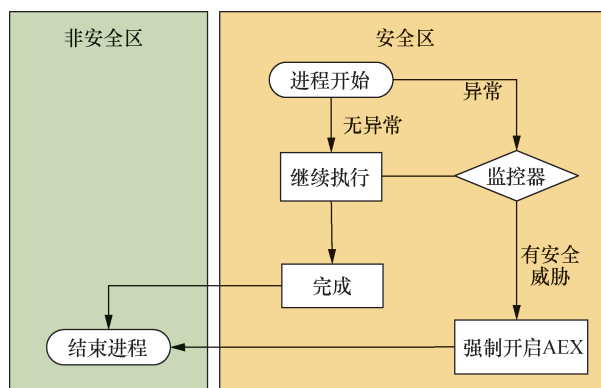


图9 SGX 监控模式流程  
Figure 9 Flow chart of SGX monitoring mode

## 5.2 SGX 性能提升

随着 SGX v2 对性能的提升以及广大研究人员不断开发出降低性能损耗的技术, 基于 SGX 开发的应用在性能上的消耗减少了很多。但对于需要进行安全区和非安全区多次交互的程序, 为了保证安全性, 需要对页面进行完整性和机密性保障, 这导致 SGX 的模式切换消耗依然很高。SGX 现阶段的密钥管理同样面临着性能问题, 即较为复杂的场景中实用性较低, 未来需要更高效的解决方案。当 SGX 作为可信计算基在对系统或软件进行完整性校验、对敏感数据进行存取和安全区内代码运行时都会产生大量的性能消耗, 如果这些性能消耗问题无法得到优化, 那么用户体验会受到较为明显的影响。可以通过优化加密内存页管理或改良页面完整性校验方法等手段减少性能消耗。

## 5.3 SGX 信任问题

SGX 技术自诞生以来, 自身设计和对 Intel ME 一些代码模块的依赖以及处理器架构产生的漏洞对安全性的影响, 让用户无法完全信任 SGX 技术。并且, SGX 远程认证功能的实现需要以信任 Intel 为前提, 依赖于 Intel IAS 进行远程认证, 但用户无法对 ISA 进行监管, 这使用户对 SGX 的可信性和安全区内数据的安全性会产生怀疑。虽然在后续工作中英特尔通过引入新特性 FLC (flexible launch control) 开放了第三方证明服务, 开始支持第三方的加载和远程认证技术, 使开发者可以脱离 Intel IAS 管控的相关限制。但固件的不可审计依然令很多用户感到担忧, 现阶段的 SGX 无法让用户应对硬件厂商在出售设备时植入恶意代码的情况, 未来可以针对固件建立机密性、完整性和可用性的安全性评估, 如固件中是否包含需要保护的秘密 (如 BIOS 密码), 系统上运行的固件是否是正确的、最新的固件, 是否被破坏或修改; 安全区是否可以访问其固件并正常运行, 根据评估完成对固件的安全审计, 增强用户对 SGX 的信任度。

## 5.4 SGX 与可信计算

将 SGX 与可信计算技术结合来达到提升安全性目的的研究工作, 自 SGX 被提出以来就在不断地进行。SGX 技术相比其他可信计算技术, 除了具有安全存储的功能, 更重要的是可以从硬件层面对软件运行时的环境进行隔离, 保障软件运行态的安全。如果将这一功能应用到可信环境的构建, 把 SGX 与其他可信计算技术相结合, 能够达到保障可信应用运行态安全性的目的。例如, 将 TPM 与 SGX 技术进行结合, 从而构建更安全的可信计算环境的设计已经出现, 正如上文中提到的 TGX。虽然将 SGX 技术与 TPM 结合起来提高安全性的想法在设计上成功, 但是想要实际投入使用还有很多问题需要考虑, 如接口层面和数据流传输过程中信息的安全性如何保障、引入其他可信计算技术对本就较为严重的性能消耗问题该如何解决, 以及将 SGX 与可信计算技术结合产生更大的攻击面会不会对安全性产生影响。

SGX 相比其他可信计算技术在可信启动领

域具有更大的优势。作为保护主机安全的重要手段,可信计算利用物理平台上的可信芯片作为物理可信根,在系统引导阶段根据度量配置策略,在系统启动前对系统进行度量,然后通过比较度量值与基准值是否相等来判断当前主机是否可信。当 SGX 作为物理可信根应用在可信启动阶段时,SGX 局部可信的机制和集成在 CPU 中的便捷性让它相比传统可信计算技术更具优势。传统的可信启动使用的物理可信根的 TCB 较大,效率也较低;而 SGX 不要求整个主机是可信的,只要求 SGX 所管理的安全区可信。基于 SGX 实现的可信启动可分为硬件平台的引导阶段和操作系统的启动阶段两部分。硬件平台的引导阶段从平台加电开始,到 BIOS 将控制权交给 Boot 之前结束。在硬件平台的引导期要保证 SGX 的可信性。操作系统的启动阶段从主引导区调入操作系统装载程序开始,到操作系统内核运行完毕并运行 Init 进程之前。该阶段需要使用 SGX 技术保证系统启动过程的可信性。

以 SGX 作为可信根可以构建可信安全计算框架,为其他领域提供可信计算支持。SGX 具有可以保障数据的安全存储和软件运行态安全的特点,使数据代码的完整性和保密性能够得到芯片级的安全保障,且允许用户远程对这些安全保护进行验证。百度在此基础上将 SGX 和 FaaS 云计算模型的优点进行有机结合,提出了全球首个通用安全计算平台 MesaTEE,并增加了内存安全保障和高兼容性支持,使金融、自动驾驶和医疗等安全敏感服务能够安全地在远程平台上处理数据,通过 SGX 为人工智能、大数据分析、区块链以及机器学习等领域提供可信安全计算平台,MesaTEE 在多个领域具有较好的应用前景。MesaTEE 作为下一代可信安全计算框架,在未来仍需要研究人员共同努力,丰富 SGX 技术生态,为用户提供更多的功能。

## 6 结束语

本文首先介绍了 SGX 技术的产生背景和相关机制,然后回顾了近年来学术界和工业界在可信计算研究领域通过 SGX 技术增强系统、应用程序和脚本语言的安全性的主要成果,对其性能、

安全性和开发及应用方面的研究瓶颈和对应的解决方法进行了详细研究,最后通过分析现有研究指明了未来研究的趋势。本文对 SGX 技术在可信计算领域的研究与分析进行了深入分析和总结。期望通过这项工作,能够给以后的研究者提供有益的借鉴与参考,为 SGX 技术未来的应用和发展做出贡献。

## 参考文献:

- [1] Intel Corporation. Intel software guard extensions (Intel SGX). Intel Labs[EB].
- [2] BAUMANN A, PEINADO M, HUNT G C, et al. Shielding applications from an untrusted cloud with Haven[C]//Operating systems Design and Implementation, 2014: 267-283
- [3] ARNAUTOV S, TRACH B, GREGOR F, et al. SCONE: secure Linux containers with Intel SGX[C]//Proc of the 12th USENIX Symp. on Operating Systems Design and Implementation(OSDI). 2016.
- [4] SCHUSTER F, COSTA M, FOURNET C, et al. VC3: trustworthy data analytics in the cloud using SGX[C]//IEEE Symposium on Security & Privacy. 2015.
- [5] SHIH M W, KUMAR M, KIM T, et al. S-NFV: securing NFV states by using SGX[C]//Proc of the 2016 ACM Int'l Workshop on Security in Software Defined Networks & Network Function Virtualization. 2016: 45-48.
- [6] ANT GROUP. Confidential computing cloud service[EB].
- [7] Intel Corporation. Intel software guard extensions developer guide[EB].
- [8] DHIMAN C, ATUL A J, SVEN B. Poster: TGX: secure SGX enclave management using TPM[C]//Proc. of the 2017 Network and Distributed System Security Symp (NDSS 2019). 2019.
- [9] 严飞, 于钊, 张立强, 等. vTSE: 一种基于 SGX 的 vTPM 安全增强方案[J]. 工程科学与技术, 2017(2).  
YAN F, YU Z, ZHANG L Q, et al. vTSE: a safety enhancement scheme of vTPM based on SGX [J]. Engineering Science and Technology, 2017(2).
- [10] 百度安全. MesaTEE 安全计算平台[EB].  
Baidu security. Mesatee security computing platform[EB].
- [11] VISHAL K, ERICK B, LIN Z Q, et al. 2017. SGX-Log: securing system logs with SGX[C]//Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security (ASIA CCS '17. Association for Computing Machinery. 2017: 19-30.
- [12] SINISA M, KARL W, MORITZ S, et al. BITE: bitcoin lightweight client privacy using trusted execution[C]//Proceedings of the 28th USENIX Conference on Security Symposium (SEC'19). 2019: 783-800.
- [13] PRIEBE C, VASWANI K, COSTA M. EnclaveDB: a secure data-

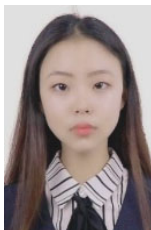
- base using SGX[C]//2018 IEEE Symposium on Security and Privacy (SP). 2018.
- [14] BZUMAN E, LIN Z Q. 2016. A case for protecting computer games with SGX[C]//Proceedings of the 1st Workshop on System Software for Trusted Execution (SysTEX '16). Association for Computing Machinery. 2016: 1-6.
- [15] GOLTZSCHE D, WULF C, MUTHUKUMARAN D, et al. TrustJS: trusted client-side execution of Javascript[C]//The 10th European Workshop. 2017.
- [16] 冯达, 王强, 赵译文, 等. 基于 SGX 的证书可信性验证与软件安全签发系统[J]. 信息安全, 2018(3): 63-69.
- FENG D, WANG Q, ZHAO Y W, et al. Certificate credibility verification and software security issuing system based on SGX[J]. Information Network Security, 2018(3): 63-69.
- [17] OLGA O, FELIX S, CEDRIC F, et al. 2016. Oblivious multi-party machine learning on trusted processors[C]//Proceedings of the 25th USENIX Conference on Security Symposium(SEC'16). 2016: 619-636.
- [18] WANG H B, ERICK B, VIISHAL K, et al. 2019. Running language interpreters inside SGX: a lightweight, legacy-compatible script code hardening approach[C]//Proceedings of the 2019 ACM Asia Conference on Computer and Communications Security (Asia CCS '19). Association for Computing Machinery. 2019: 114-121.
- [19] 王鹏, 樊成阳, 程越强, 等. SGX 技术的分析和研究[J]. 软件学报, 2018, 9(9): 2778-2798.
- WANG J, FAN C Y, CENG Y Q, et al. Analysis and research of SGX technology[J]. Journal of Software, 2018, 9(9): 2778-2798.
- [20] WEISSE O, BERTACCO V, AUSTIN T. Regaining lost cycles with hotcalls: a fast interface for SGX secure enclaves[C]//The 44th Annual International Symposium. 2017.
- [21] TIAN H, ZHANG Y, XING C, et al. SGX Kernel: a library operating system optimized for intel SGX[C]//The Computing Frontiers Conference. 2017.
- [22] SARTAKOV V A, BRENNER S, MOKHTAR S B, et al. EActors: fast and flexible trusted computing using SGX[C]//The 19th International Middleware Conference. 2018.
- [23] XING B C, SHANAHAN M, LESLIE H R. Intel software guard Extensions (Intel SGX) software support for dynamic memory allocation inside an enclave[C]//Proc of the Hardware and Architectural Support for Security and Privacy 2016 (HASP 2016). 2016: 1-9.
- [24] ORENBACH M, LIFSHITS P, MINKIN M, et al. Eleos: ExitLess OS services for SGX enclaves[C]//Proc of the 12th European Conf on Computer Systems. 2017: 238-253.
- [25] MORITZ L, MICHAEL S, DANIEL G, et al. Meltdown: reading kernel memory from user space[C]//USENIX Security Symposium. 2018.
- [26] KOCHER P, GENKIN D, GRUSS D, et al. Spectre Attacks: Exploiting speculative execution[J]. Communications of the ACM, 2018, 63(7).
- [27] JO V B, MARINA M, OFIR W, et al. Foreshadow: extracting the keys to the Intel SGX kingdom with transient out-of-order execution[C]//USENIX Security Symposium. 2018.
- [28] SCHAIK S V, MILBURN A, STERLUND S, et al. RIDL: rogue in-flight data load[C]//Symposium on Security & Privacy. 2019.
- [29] OLEKSII O, BOHDAN T, ROBERT K, et al. Varys: protecting SGX enclaves from practical side-channel attacks[C]//Proceedings of the 2018 USENIX Conference on Usenix Annual Technical Conference (USENIX ATC '18). 2018: 227-239.
- [30] MURDOCK K, OSWALD D, GARCIA F D, et al. Plundervolt: software-based fault injection attacks against intel SGX[C]//2020 IEEE Symposium on Security and Privacy(SP). 2020.
- [31] MUSTAKIMUR R K, CHENG Y Q, WANG Z, et al. 2020. Coin attacks: on insecurity of enclave untrusted interfaces in SGX[C]//Proceedings of the 25 International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLoS '20). Association for Computing Machinery. 2020: 971-985.
- [32] BAUMAN E, WANG H B, ZHANG M W, et al. 2018. SGXElide: enabling enclave code secrecy via self-modification[C]//Proceedings of the 2018 International Symposium on Code Generation and Optimization (CGO 2018). Association for Computing Machinery, 2018: 75-86.
- [33] JAIN P, DESAI S, KIM S, et al. OpenSGX: an open platform for SGX research[C]//NDSS. 2016.
- [34] CHOI C, KWAK N, JANG J, et al. S-OpenSGX: a system-level platform for exploring SGX enclave-based computing[J]. Computers & Security, 2017, 70(sep.):290-306.
- [35] DING Y, DUAN R, LI L, et al. POSTER: Rust SGX SDK: towards memory safety in Intel SGX enclave[C]//Proc of the 2017 ACM SIGSAC Conf. on Computer and Communications Security (CCS 2017). 2017: 2491-2493.
- [36] MesaPy[EB].
- [37] GHOSN A, LARUS J R, BUGNION E. Secured routines: language-based construction of trusted execution environments[C]//2019 USENIX Annual Technical Conf (ATC 2019). 2019: 571-586.
- [38] SHINDE S, TIEN D L, TOPLE S, et al. Panoply: low-TCB Linux applications with SGX enclaves[C]//Network & Distributed System Security Symposium. 2017.
- [39] TSAI C C, PORTER D E, VIJ M. Graphene-SGX: a practical library OS for unmodified applications on SGX[C]//Proc of 2017 USENIX Annual Technical Conf (ATC 2017). 2017: 645-658.
- [40] Occlum[EB].

- [41] VICTOR C, SRINIVAS D. Intel SGX Explained[C]/IACR Cryptology ePrint Archive 2016. 2016: 86.
- [42] 刘国杰, 张建标. 基于 TPCM 的服务器可信 PXE 启动方法[J]. 网络与信息安全学报, 2020, 6(6): 105-111.  
LIU G J, ZHANG J B. TPCM-based trusted PXE boot method for servers[J]. Chinese Journal of Network and Information Security, 2020, 6(6): 105-111.
- [43] RICH U, GIL N, DION R, et al. Intel virtualization technology[J]. Computer, 2005, 38(5):48-56.
- [44] 董攀, 丁滢, 江哲, 等. 基于 TEE 的主动可信 TPM/TCM 设计与实现[J]. 软件学报, 2020, 31(5): 1392-1405.  
DONG P, DING Y, JIANG Z, et al. Active and credible TPM/TCM design and implementation based on TEE[J]. Journal of Software, 2020, 31(5): 1392-1405.
- [45] 章张锴, 李舟军, 夏春和, 等. 借助 Hypervisor 强化 TrustZone 对非安全世界的监控能力[J]. 软件学报, 2018(8):2511-2526.  
ZHANG Z K, LI Z J, XIA C H, et al. Strengthening the monitoring capability of TrustZone to non-secure world with the help of Hypervisor [J]. Journal of Software, 2018(8):2511-2526.
- [46] WU Y, LIU Y, LIU R, et al. Comprehensive VM protection against untrusted hypervisor through retrofitted AMD memory encryption[C]/IEEE International Symposium on High Performance Computer Architecture. I2018.

#### [作者简介]



赵波 (1972- ), 男, 山东青岛人, 博士, 武汉大学教授、博士生导师, 主要研究方向为信息系统安全、可信计算、嵌入式安全、人工智能及大数据安全隐私保护。



袁安琪 (1997- ), 女, 山东荣成人, 武汉大学硕士生, 主要研究方向为系统安全、可信计算。



安杨 (1975- ), 女, 湖北武汉人, 博士, 武汉大学副教授, 主要研究方向为计算机应用。