

硬件辅助系统安全

陈海波 / 夏虞斌

上海交通大学并行与分布式系统研究所

<https://ipads.se.sjtu.edu.cn>

版权声明

- 本内容版权归**上海交通大学并行与分布式系统研究所**所有
- 使用者可以将全部或部分本内容免费用于非商业用途
- 使用者在使用全部或部分本内容时请注明来源：
 - 内容来自：上海交通大学并行与分布式系统研究所+材料名字
- 对于不遵守此声明或者其他违法使用本内容者，将依法保留追究权
- 本内容的发布采用 Creative Commons Attribution 4.0 License
 - 完整文本：<https://creativecommons.org/licenses/by/4.0/legalcode>

当操作系统不可信时...

为什么要假设操作系统是恶意的？

- **系统的复杂性**

- 软件：恶意软件，OS本身可能存在漏洞
- 硬件：外设越来越智能，本身可能存在漏洞，甚至是恶意构造
- 环境：云计算环境、IoT设备，面临这更复杂多变的
- 人：运维外包（如云计算等）导致接触计算机的人更复杂

- **一种更简单的威胁模型**

- “除了应用，别的都不可信”

恶意操作系统如何攻击应用？

- **应用的攻击面**
 - 同层：其他应用程序
 - 底层：操作系统、Hypervisor、硬件
- **操作系统窃取应用的数据**
 - 操作系统控制着页表，可直接映射应用的内存并读取数据
- **操作系统改变应用的执行**
 - 操作系统控制着页表，可直接在应用内部新映射一段恶意代码
 - 操作系统可任意改变程序的RIP，劫持其执行流

现实中的类似攻击

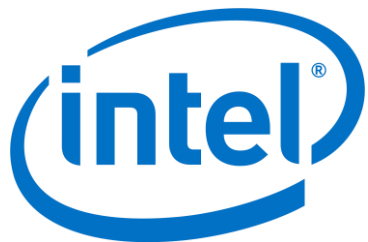
- 系统对应用的攻击

- 攻击者获取具有root权限的bash（如利用sticky位程序的漏洞）
- 攻击者获得`insmod`的权限，在内核中插入恶意的模块
- 攻击者篡改内核的文件，下次启动后加载
- 攻击者获得`kexec`的权限，动态执行另一个内核
- 攻击者获得hypervisor的权限，从更底层发起攻击
- 攻击者能够控制某个设备（如智能网卡），直接访问物理内存
-

一种新的威胁模型：硬件Enclave

- **不信任CPU外的硬件**
 - 包括内存（DRAM）、设备、网络
- **仅信任CPU**
 - 包括cache、所有计算逻辑（Anyway，总得信任CPU吧...）
- **Enclave（飞地）**
 - 又称为可信执行环境，TEE（Trusted Execution Environment）
 - 什么是“可信”？信什么呢？

工业界主流的Enclave系统



- Intel SGX



- AMD SEV



- ARM TrustZone

ARM TrustZone的应用

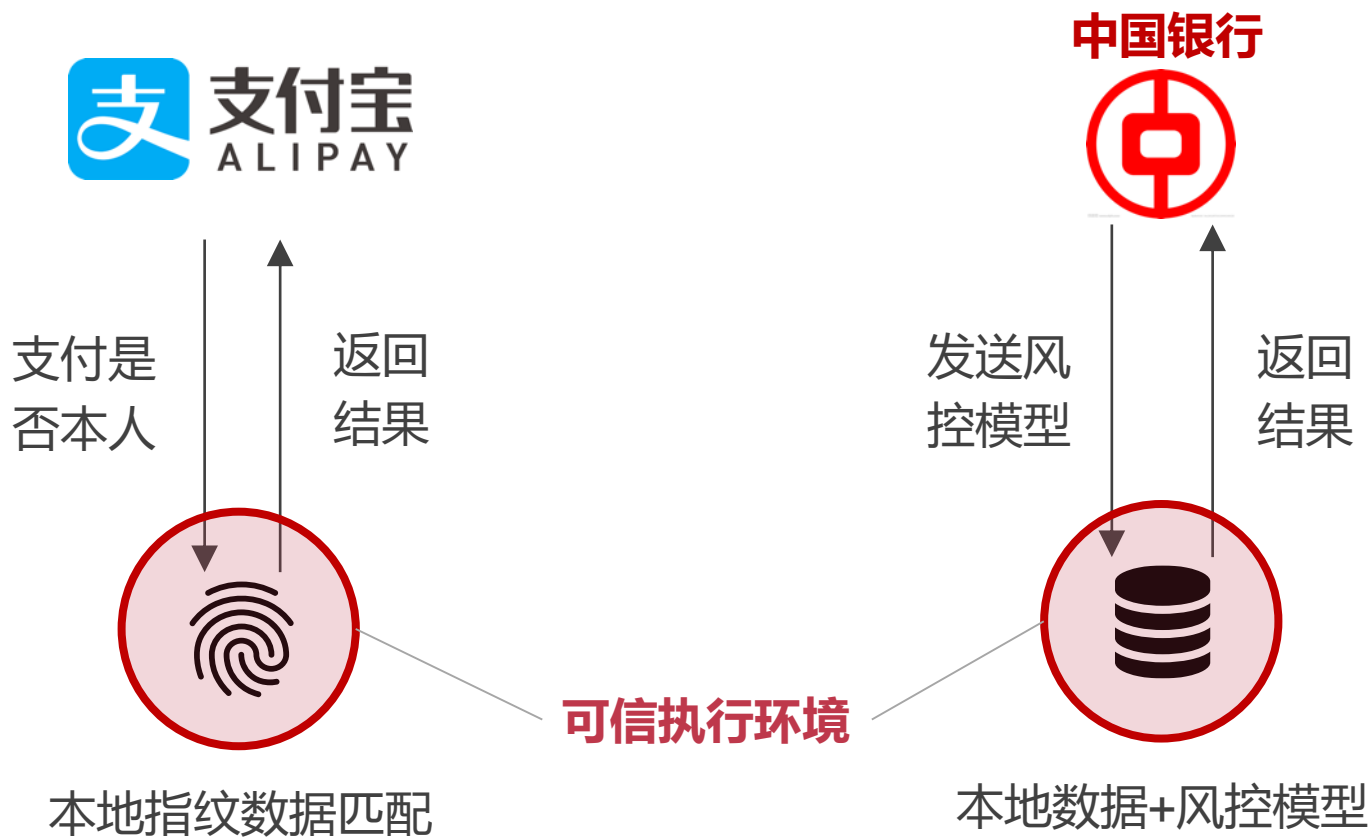
- TEE已经成为**生物识别**设备的标配
 - 使用TEE来隔离指纹的采集、存储、验证等过程
 - 即使手机被越狱或Root，攻击者也无法获取指纹数据



数据安全流通的使能技术之一

- **数据需要流动才能产生更大的价值**
 - 从单维到多维，从小数据到大数据
 - 通过数据融合、共享，挖掘出新的价值
- **数据太容易复制，导致不可控**
 - 一旦数据被共享，则极有可能导致数据沉淀
 - 例如：识别身份证的代理，最后自己提供识别服务
 - TEE可保证数据在使用过程中的安全性
 - 原理：数据明文仅在Enclave中出现，Enclave之外皆为密文

数据安全管控：以指纹为例



利用Enclave提供的信任打通多个机构

2019年8月，多家公司成立机密计算联盟



Security



Intel, Google, Microsoft, and others launch
Confidential Computing Consortium for data
security

KHARI JOHNSON @KHARIJOHNSON AUGUST 21, 2019 11:55 AM



CONFIDENTIAL COMPUTING
CONSORTIUM

VB TRANSFORM

The AI event for
business leaders

Hosted Online
July 15 - 17

Learn More

<https://venturebeat.com/2019/08/21/intel-google-microsoft-and-others-launch-confidential-computing-consortium-for-data-security/>

保护Enclave的方法

- **基于隔离**
 - 使操作系统没有权限访问用户的数据
- **基于加密**
 - 操作系统即使访问用户数据，也无法解密
- **基于隔离+加密**
 - 隔离防御软件攻击，加密防御硬件攻击

隔离的方法

- **基于预留的隔离（硬件）**
 - 例如：PRM（Processor Reserved Memory）
 - CPU预留一部分物理内存，不提供给操作系统
- **基于页表的隔离（操作系统）**
 - 例如：保证操作系统无法映射应用的物理内存页
 - 问题：页表是由操作系统自己管理的，监守自盗？
- **基于插桩的隔离（编译器）**
 - 例如：SFI（Software Fault Isolation）
 - 在每次访存前插入边界检查，性能损失较大

隔离的优缺点

- **优点:**

- 现有系统本来就需要隔离，对逻辑影响较小
- 基于硬件检查的隔离对性能的影响较小

- **缺点:**

- 通常需要硬件的支持，例如引入新的权限层
- 很难保证隔离的完整性，需要信任设计与实现
 - 例如：CPU通过页表实现的隔离，对恶意设备来说无效

案例：INTEL SGX

Intel SGX

- **SGX: Software Guard eXtension**
 - 2015年首次引入Intel Skylake架构
 - 保护程序和代码在运行时的安全 (data in-run)
 - 其他安全包括：存储时安全和传输时安全
- **关键技术**
 - Enclave内部与外部的隔离
 - 内存加密与完整性保护
 - 远程验证

硬件内存加密与保护机制

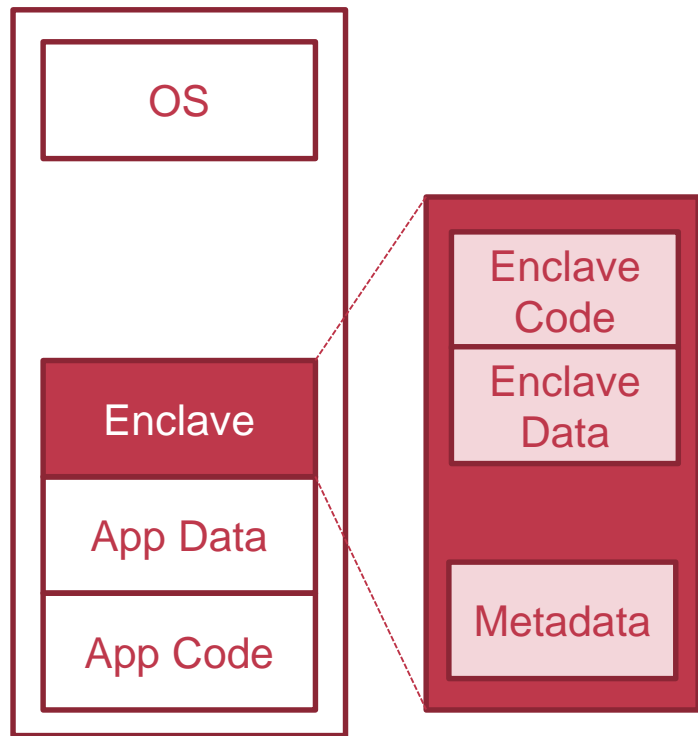
- **硬件加密保护隐私性**
 - CPU外皆为密文，包括内存、存储、网络等
 - CPU内部为明文，包括各级Cache与寄存器
 - 数据进出CPU时，由进行加密和解密操作
- **硬件Merkle Tree保护完整性**
 - 对内存中数据计算一级hash，对一级hash计算二级hash，形成树
 - CPU内部仅保存root hash，其它hash保存在不可信的内存中
 - 当内存中的数据被修改时，更新Merkle Tree

EPC (Enclave Page Cache)

- **CPU预留一部分内存，仅允许Enclave访问**
 - 连续的128MB/256MB，这部分不会暴露给软件操作
 - 全部加密，并保证数据的完整性（即无法篡改）
 - 可能的篡改方法：通过总线直接修改
- **操作系统负责将EPC映射至Enclave中**
 - 操作系统也可以触发swap，将数据从EPC交换到DRAM中
 - 由专门的硬件指令来进行swap，粒度为一个内存页（4K）
 - 注意：页表依然由不可信的操作系统控制

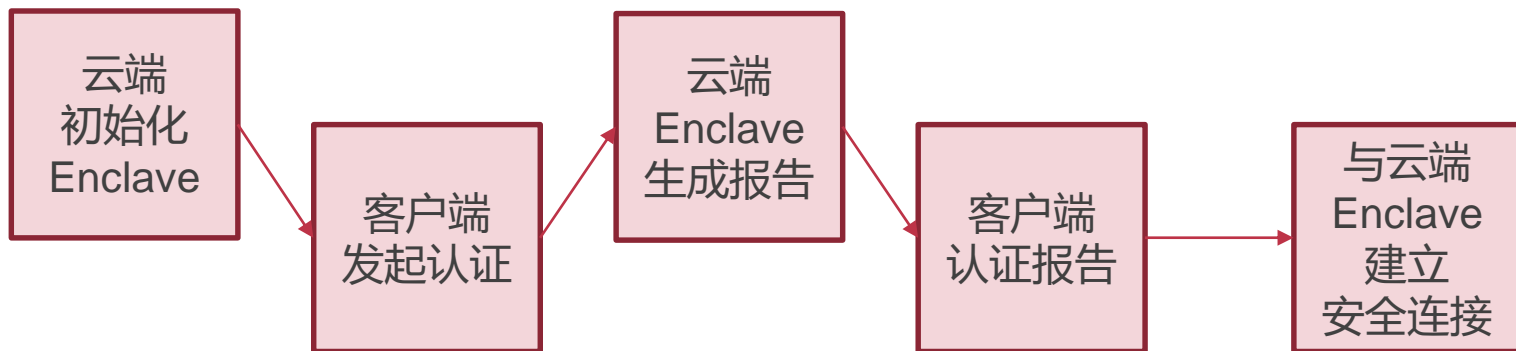
Enclave与进程的关系

- **Enclave是进程的一部分**
 - Enclave内外共享一个虚拟地址空间
 - Enclave内部可以访问外部的内存
 - 反之则不行
- **创建Enclave的过程**
 1. OS创建进程
 2. OS分配虚拟地址空间
 3. OS将Enclave的code加载到EPC中
 - 并将EPC映射到Enclave的虚拟地址
 - 循环3，完成所有code加载和映射
 4. 完成进程创建



远程验证 (Remote Attestation)

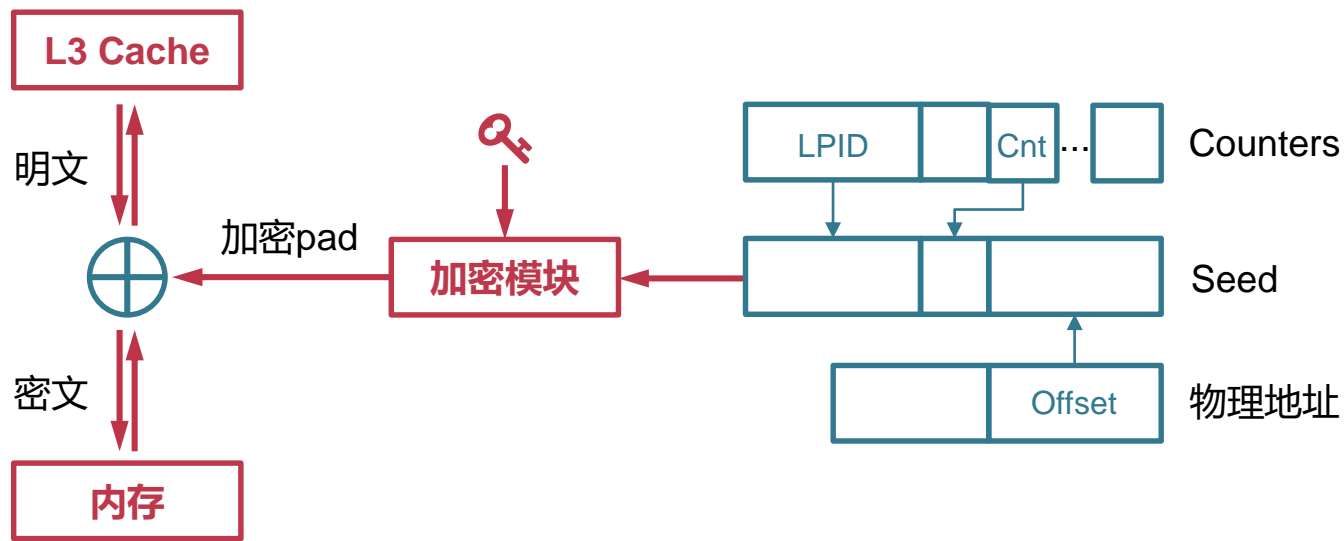
- 要解决的问题：如何远程判断某个主体是Enclave？
 - 例如，如何判断某个在云端的服务运行环境是安全的
 - 必须在认证之后，再进行下一步的操作，例如发送数据



硬件内存加密

- **加密的最小单位**
 - Cache line: 64Byte (512bit)
- **方法一：单密钥加密**
 - 缺点：同样的明文会产生同样的密文
- **方法二：多密钥加密**
 - 缺点：如何保存这些密钥？CPU内部放不下
- **方法三：单密钥 + 多 seed**
 - 为每个cache line单独生成一个seed，用密钥加密后，对数据进行异或

生成seed用于加密



硬件内存加密

- **采用加密pad的方式而不是直接加密数据**
 - 每个内存区域对应一个pad
 - 对pad进行加密，然后数据与pad进行异或以实现加解密
- **加密pad由key和seed共同组成**
 - Seed可以由时空因子组成
 - 明文只需一次异或运算得到密文
- **任意两次加密的pad必须不同**
 - 如果两次加密的pad相同，可以轻松的反推明文

内存完整性保护

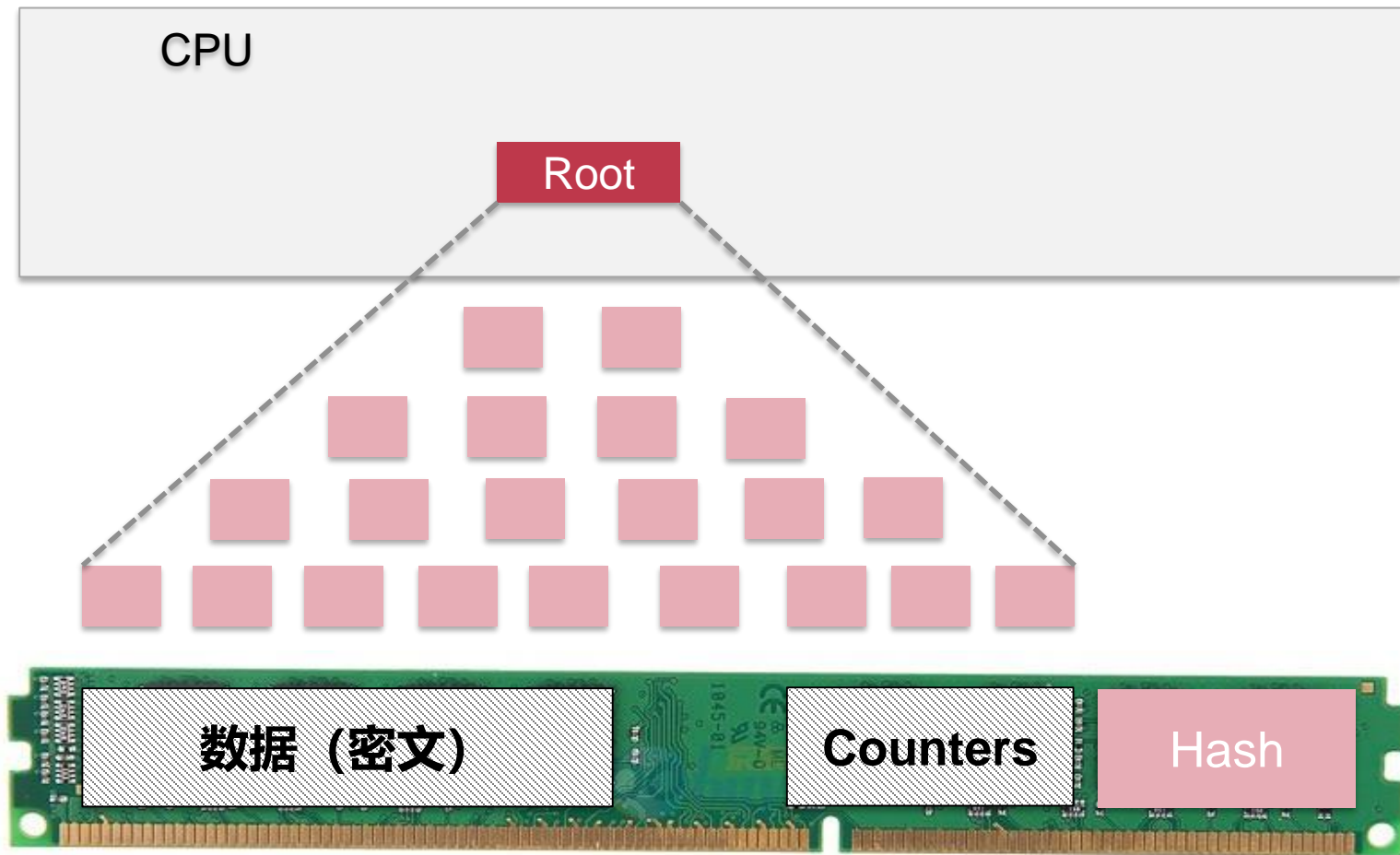
- **Merkle hash Tree**

- 可以保证内存不会受到拼接和欺骗攻击
 - 不知道hash key无法计算对应的mac
- 无法防御回放攻击
 - 攻击者可以将mac和data同时替换成老版本

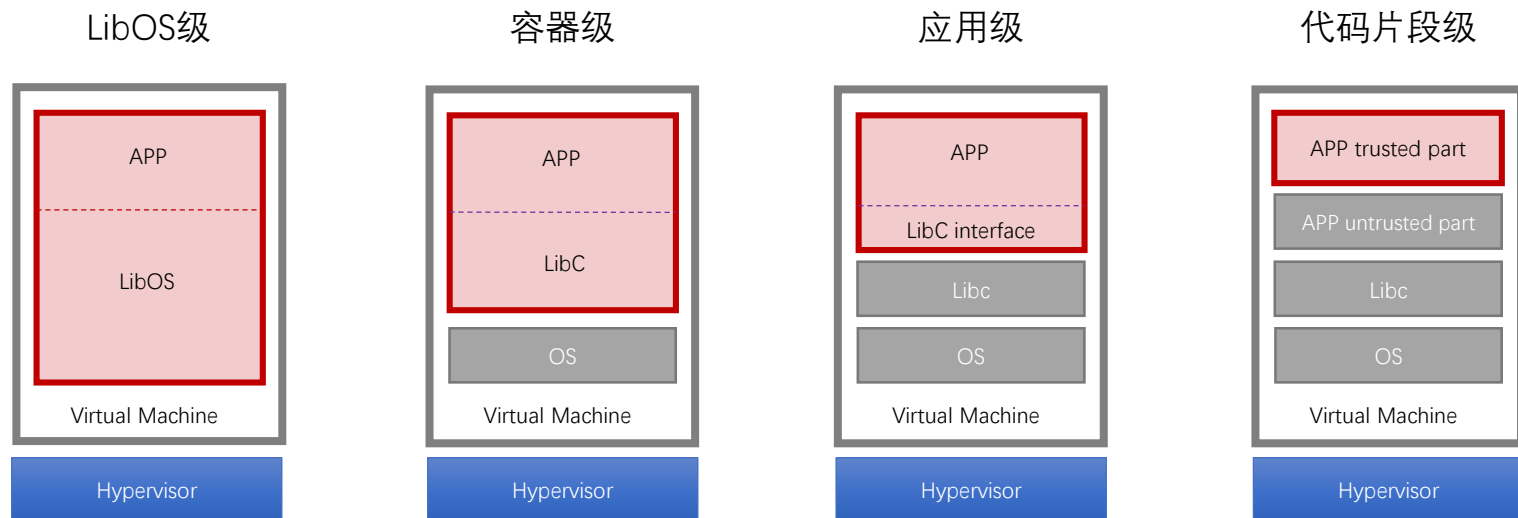
- **将root hash (mac) 存储在CPU中**

- 防御回放攻击
 - 攻击者无法修改root mac的值

内存完整性保护：Merkle Hash Tree



基于Intel SGX的软件架构

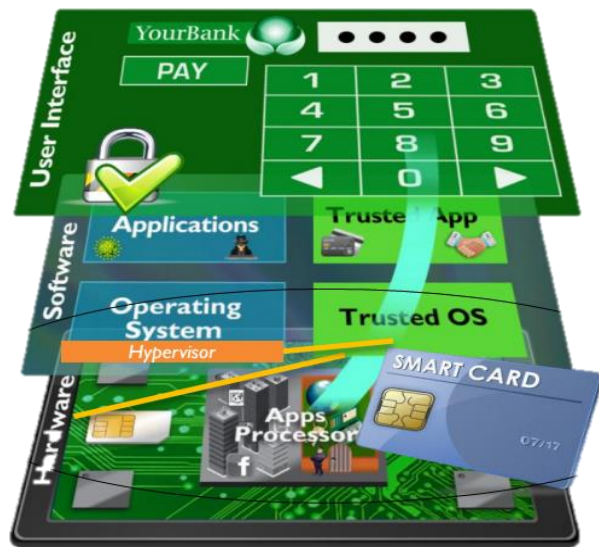


	应用兼容	TCB 大小	Ocall 数量	攻击面	保护OS
LibOS级	部分	大	少	小	✓
容器级	✓	中	中	中	✗
应用级	✓	中	多	大	✗
代码片段级	✗	小	少	小	✗

其他平台的 ENCLAVE/TEE

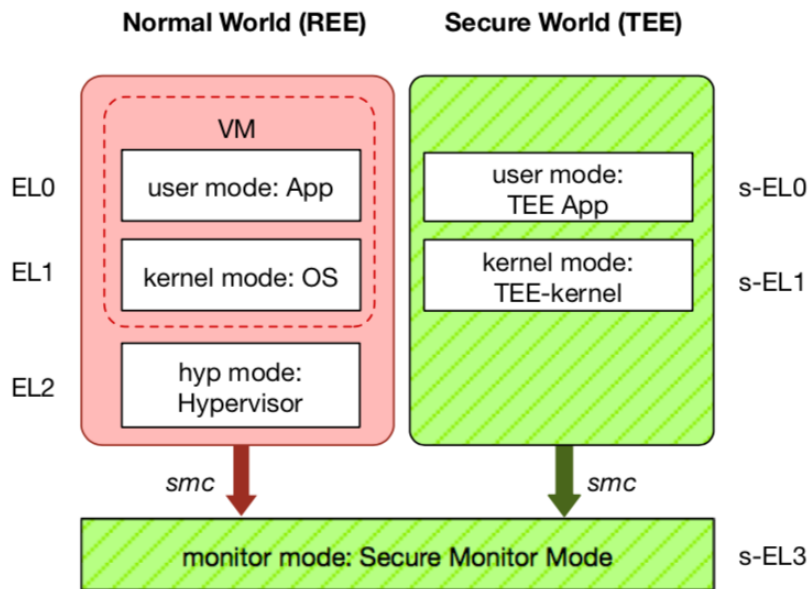
ARM TrustZone 技术

- ARMv6版本开始的安全硬件特性
 - 包括ARM11及Cortex A系列
 - 目前大部分手机芯片均有该硬件特性
- 同时运行一个安全的OS和一个普通的OS
 - 两个系统之间互相隔离运行
 - 安全的OS具有更多的权限
- TrustZone是一个全系统级别的安全架构
 - 处理器、内存和外设的安全隔离

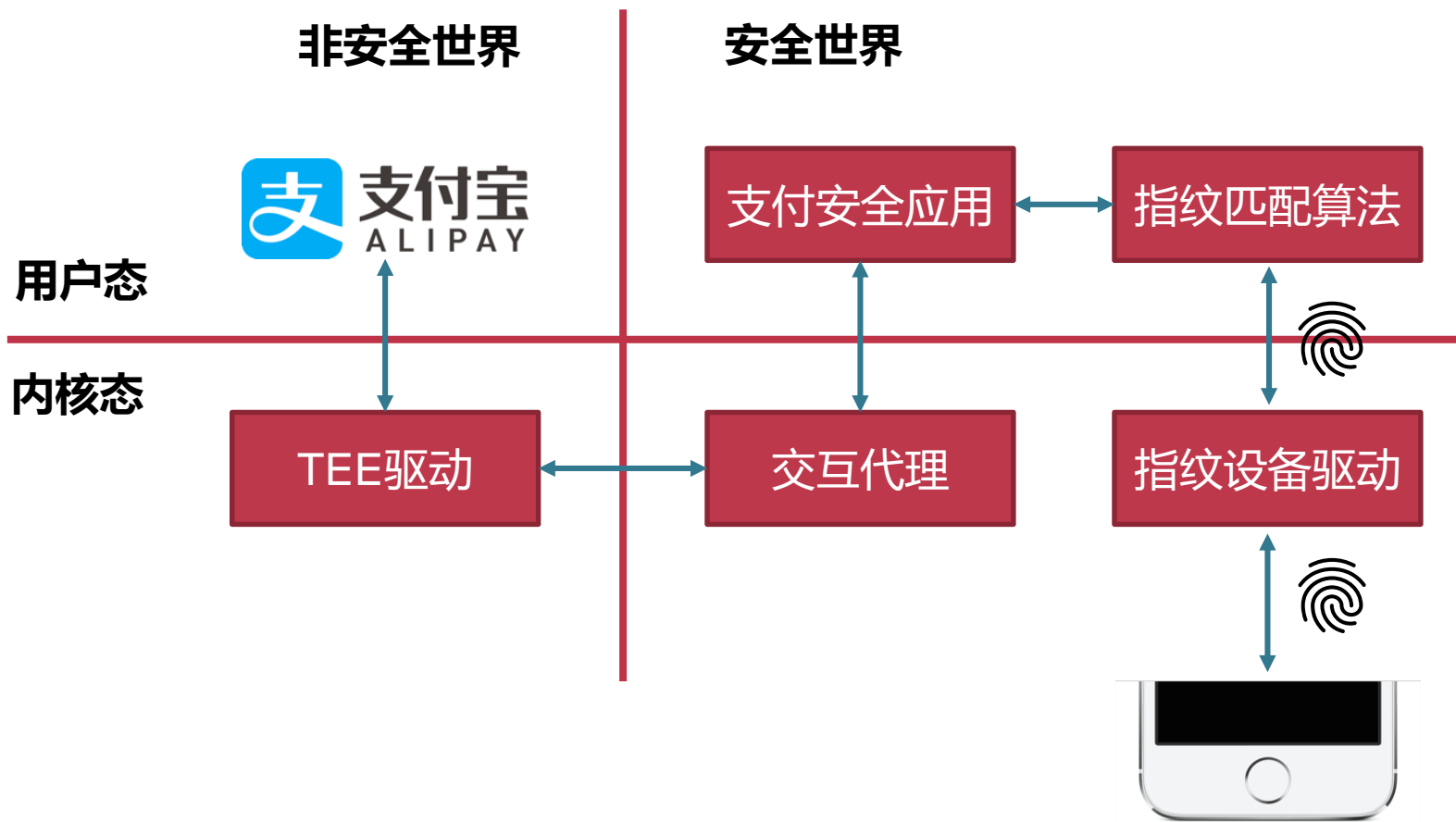


TEE硬件状态与软件架构

- **TEE内部运行一个完整的操作系统**
 - 与Android版本无关，也无需适配
 - TEE与Android通过共享内存进行交互
- **TEE内部也分内核态与用户态**
 - TEE用户态可运行多个安全应用（TA）
 - 安全应用可支持动态下载和动态更新
- **TEE内部结构**
 - Secure OS + 中间件 + 安全应用 + 外部交互



用TrustZone保护指纹的录入和识别



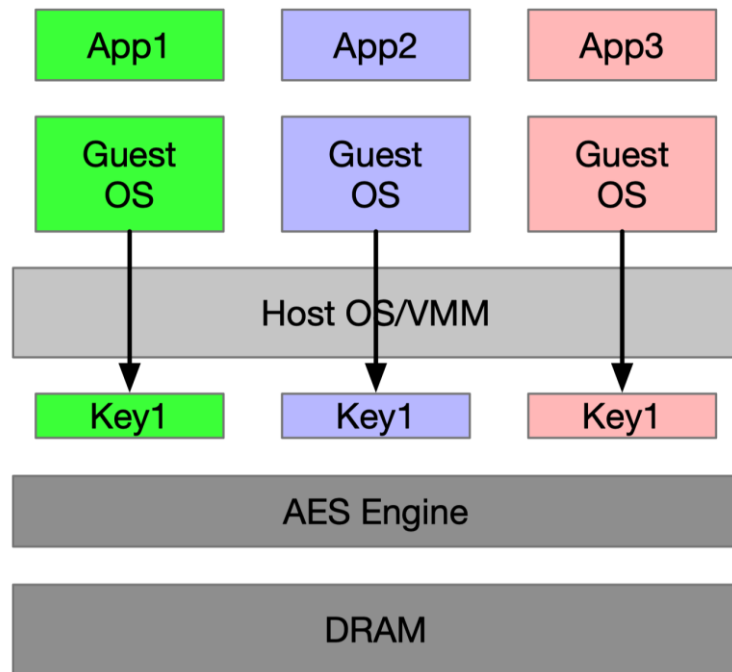
AMD SEV

- **以虚拟机为粒度的Enclave**

- 对不同的虚拟机进行加密
- 每个虚拟机的密钥均不相同
- Hypervisor有自己的密钥

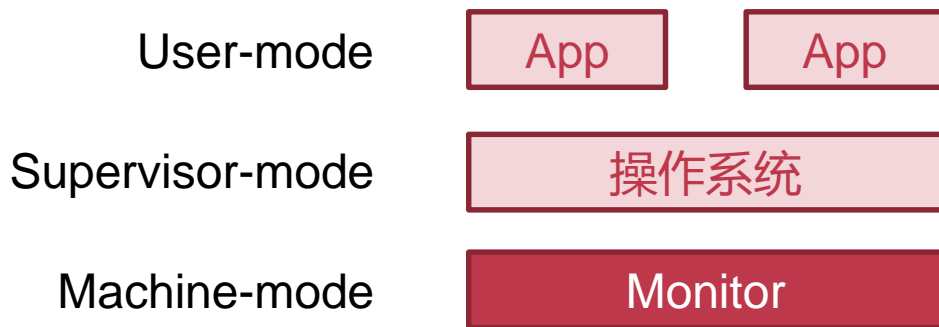
- **安全模型的缺陷**

- 依然部分依赖Hypervisor
 - 如：为VM设置正确的密钥

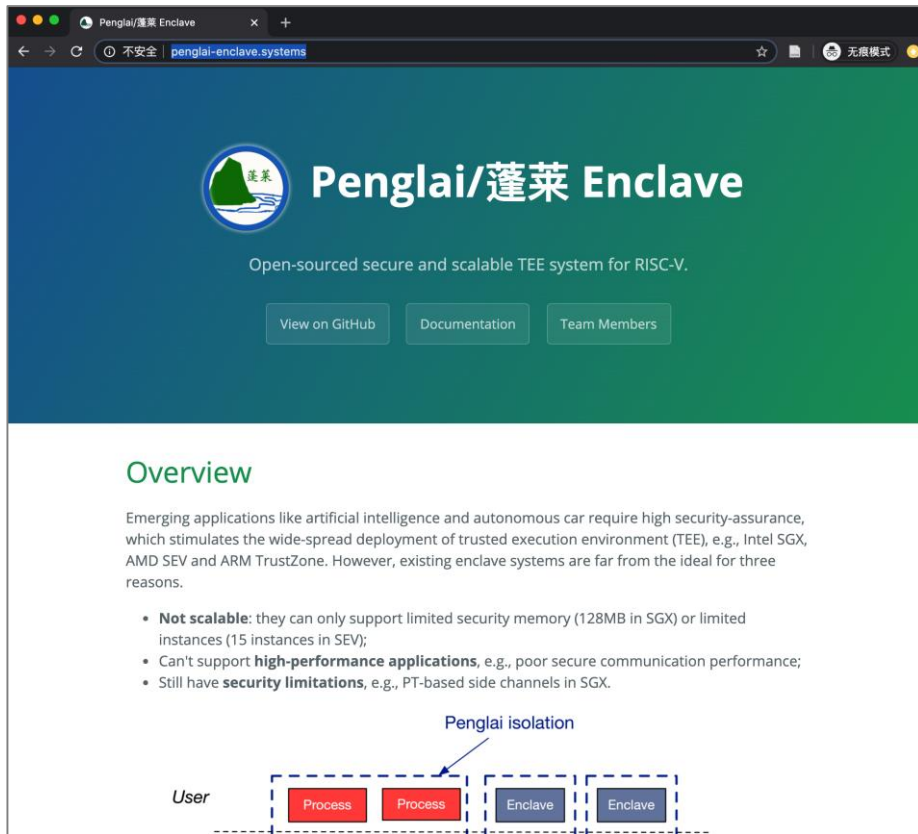


RISC-V平台的Enclave

- RISC-V具有一个新的模式：Machine-Mode
 - 位于操作系统和Hypervisor之下，直接访问物理地址
 - 具有最高权限，可访问所有的计算资源，并提供新的功能
 - 在M-Mode下实现的软件monitor，可实现Enclave的接口



我们自己的Enclave



The screenshot shows the homepage of the Penglai/蓬莱 Enclave project. The header features the project logo and name, followed by the tagline "Open-sourced secure and scalable TEE system for RISC-V." Below this are three buttons: "View on GitHub", "Documentation", and "Team Members". The "Overview" section describes the need for high security-assurance in emerging applications like AI and autonomous cars, contrasting existing systems (Intel SGX, AMD SEV, ARM TrustZone) with the ideal. It lists three reasons why existing systems are not ideal: limited security memory, poor secure communication performance, and security limitations. At the bottom, a diagram labeled "Penglai isolation" shows a "User" box containing two "Process" boxes and two "Enclave" boxes, with arrows indicating the flow of data and isolation between them.

Penglai/蓬莱 Enclave

Open-sourced secure and scalable TEE system for RISC-V.

[View on GitHub](#) [Documentation](#) [Team Members](#)

Overview

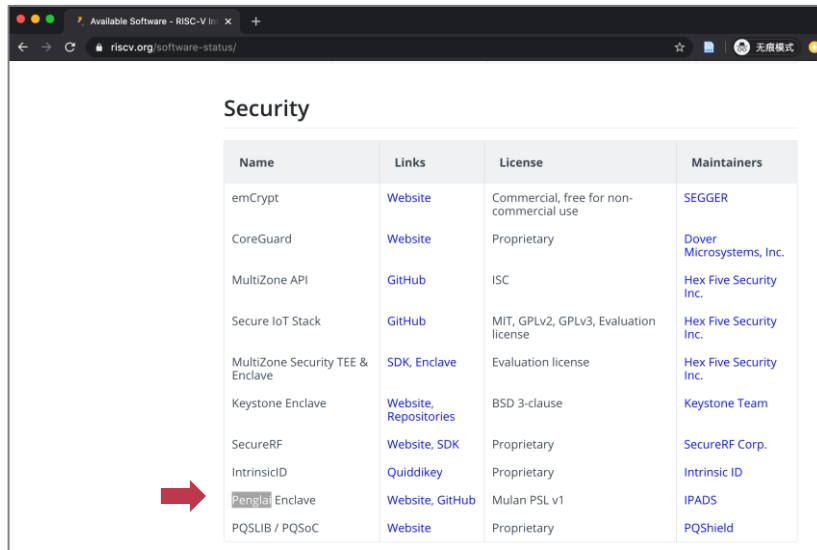
Emerging applications like artificial intelligence and autonomous car require high security-assurance, which stimulates the wide-spread deployment of trusted execution environment (TEE), e.g., Intel SGX, AMD SEV and ARM TrustZone. However, existing enclave systems are far from the ideal for three reasons.

- **Not scalable:** they can only support limited security memory (128MB in SGX) or limited instances (15 instances in SEV);
- Can't support **high-performance applications**, e.g., poor secure communication performance;
- Still have **security limitations**, e.g., PT-based side channels in SGX.

Penglai isolation

User

Process Process Enclave Enclave



The screenshot shows the "Security" section of the riscv.org software-status page. It contains a table with columns for Name, Links, License, and Maintainers. A red arrow points to the "Penglai Enclave" entry in the table.

Name	Links	License	Maintainers
emCrypt	Website	Commercial, free for non-commercial use	SEGGER
CoreGuard	Website	Proprietary	Dover Microsystems, Inc.
MultiZone API	GitHub	ISC	Hex Five Security Inc.
Secure IoT Stack	GitHub	MIT, GPLv2, GPLv3, Evaluation license	Hex Five Security Inc.
MultiZone Security TEE & Enclave	SDK , Enclave	Evaluation license	Hex Five Security Inc.
Keystone Enclave	Website , Repositories	BSD 3-clause	Keystone Team
SecureRF	Website , SDK	Proprietary	SecureRF Corp.
IntrinsicID	Quiddikey	Proprietary	Intrinsic ID
Penglai Enclave	Website , GitHub	Mulan PSL v1	IPADS
PQSLIB / PQSoC	Website	Proprietary	PQShield



<http://penglai-enclave.systems/>



小结

控制系统复杂性

- **Enclave的抽象是一种简化**
 - 对威胁模型和信任关系的简化
 - 例如：Intel SGX将对软硬件环境的信任规约到对Intel的信任
 - 这种简化有可能带来新的问题：Single-point of Failure
- **Enclave的主要技术**
 - 保护技术：基于权限的隔离与基于加密的控制
 - 远程验证：对密钥的管理

Enclave的不足

- **仅靠隔离是不够的，还需要考虑交互安全**
 - Enclave依然需要OS提供服务：调度、系统调用、资源分配...
 - 即使隔离，OS依然可能发起的攻击包括
 - 接口攻击：合法的系统调用返回错误的值
 - 例：malloc返回指向栈的地址，导致内部自己破坏掉栈
 - DoS攻击：拒绝分配计算资源（恶意调度）
- **依然受到侧信道等攻击的威胁**
 - Spectre、L1TF

下次课内容

- 操作系统调试