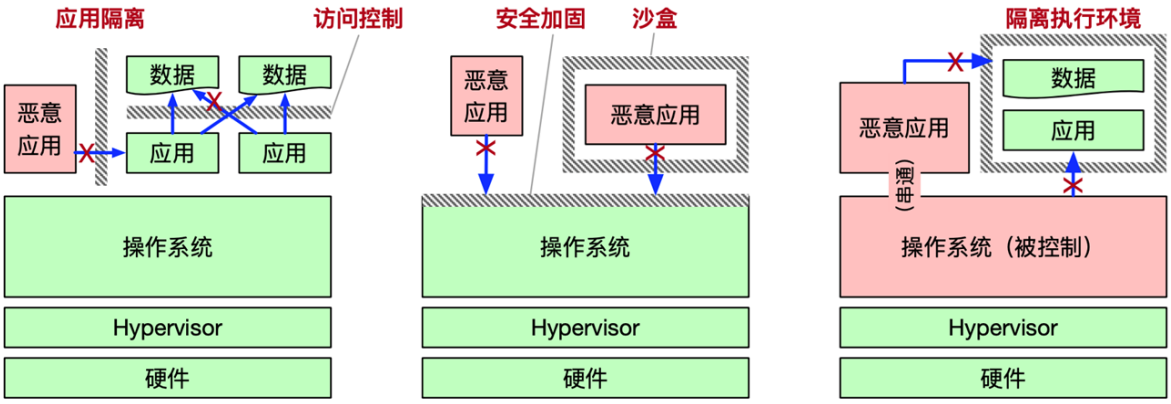
Lecture27 Hardware Security

1. 当操作系统不再可信时...



(a) 基于操作系统的
应用隔离与访问控制

(b) 操作系统对恶意
应用的隔离与防御

(c) 操作系统不可信时
对应用的保护

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：云计算设备、IoT设备，面临更多复杂多变的情况
  - 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

一种新的威胁模型：安全处理器

- 不信任CPU外的硬件

  > 包括内存(DRAM)、设备、网络

- 仅信任CPU

  > 包括cache、所有逻辑(Anyway, 总是得信任CPU吧...)

- **Enclave(飞地)**

  > 又称为可信执行环境，TEE(Trusted Execution Environment)

Enclave/TEE：可信执行环境

- **Enclave/TEE的定义**
  - Enclave，又称"可信执行环境" （TEE，Trusted Execution Environment），是计算机系统中一块通过底层软硬件构造的安全区域，通过保证加载到该区域的代码和数据的完整性和隐私性，实现对代码执行与数据资产的保护 —— *Wikipedia*
- **Enclave的两个主要功能**
  - 远程证明：验证远程节点是否为加载了合法代码的Enclave
  - 隔离执行：Enclave外无法访问Enclave内部的数据
- **Enclave带来的能力：限制访问数据的软件**
  - 可保证数据只在提前被认证的合法节点间流动
    - 合法节点：部署了合法软件的节点

2. 可信执行环境

> “可信执行环境” （TEE，Trusted Execution Environment），又称Enclave，是计算机系统中一块通过底层软硬件构造的安全区域，通过保证加载到该区域的代码和数据的完整性和隐私性，实现对代码执行与数据资产的保护

安全屋：一种”简单“的机密计算抽象

- **找一间只有一个入口和一个出口的屋子——安全屋**
  - 确保没有窗、没有通风口，没有任何可以离开的通道
  - 屋子里只有一台仅安装了训练算法的电脑
- **在出入口的地方各安装一个机械臂**
  - 十四亿人排队将照片通过U盘交给输入机械臂，输入后U盘留在房间
- **训练完成后**
  - 得到的模型存入U盘，并通过输出机械臂送到出口处
  - 输出U盘被取走后触发清理流程，屋子里的所有电子设备全部销毁

"安全屋"的实现与流程

1. **输入**
   - U盘从仅有的入口输入
2. **运算**
   - 训练输入数据生成模型
3. **输出**
   - 模型通过仅有的出口输出
4. **清理**
   - 一旦输出完成立即清理
   - 没有任何数据沉淀

安全屋面临的安全挑战

1. **如何向用户证明安全屋是真的?** ✓
2. **如何保证安全屋中的算法没有被篡改?** ✓
3. **如何保证输出后的清理过程一定会执行?** ✓
4. **如何保证安全屋的数据不被窃取或篡改?** ✓

系统层面：机密计算 → 可信执行环境

5. 如何保证算法的输出数据不包含隐私?
6. 如何保证算法的实现没有可被利用的漏洞?

算法层面

非机密计算范畴

7. 如何保证用户输入的数据不是假的或错的? → 数据层面

> 1，2两个问题对应的技术点：远程证明（在TEE中加载代码的完整性（初始时））

> 3，4两个问题对应的技术点：隔离执行（TEE中代码执行不受外部攻击（运行时））

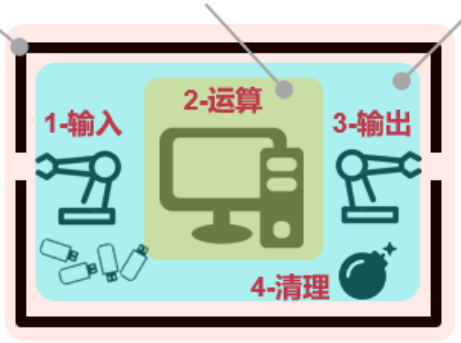
可信执行环境（TEE）的组成

TEE 硬件部分

- 通常由芯片厂商提供
- 提供两个主要能力
  - 隔离执行
  - 远程证明
- 芯片厂商预埋私钥
  - 用于远程证明

算法（不属于TEE）

- 通常由用户自己或第三方提供
- 其安全性由提供方保证



安全屋

TEE 软件部分

- 通常由云服务商提供
  - 用户也可自己提供
- 提供算法的无关通用功能
  - 输入
  - 输出
  - 初始化
  - 清理（可选）
  - ……
- 通常开源，以接收审计

3. TEE技术点一：远程证明

如何向用户证明TEE是真的?

- **认证：TEE的硬件是真的**
  - TEE来自某个芯片制造商
  - 方法：通过硬件内置的私钥来判断（对应公钥由认证服务器维护）
  - 前提：该私钥不出硬件，且对应的公钥不可伪造
- **认证：TEE的软件是真的**
  - TEE中加载的软件：包含系统软件和用户的软件
  - 方法：通过对TEE内存计算hash来判断（使用TEE内嵌私钥签名）
  - 时间节点：在所有代码加载入TEE的内存后，在执行第一行代码前

静态度量与动态度量

- **启动时的度量（静态度量）**
  - 如专用的TPM芯片（Trusted Platform Module）
  - TPM验证最早启动的软件组件（如加载器bootloader）
  - 下层软件加载验证上层软件，形成信任链，TPM是信任根

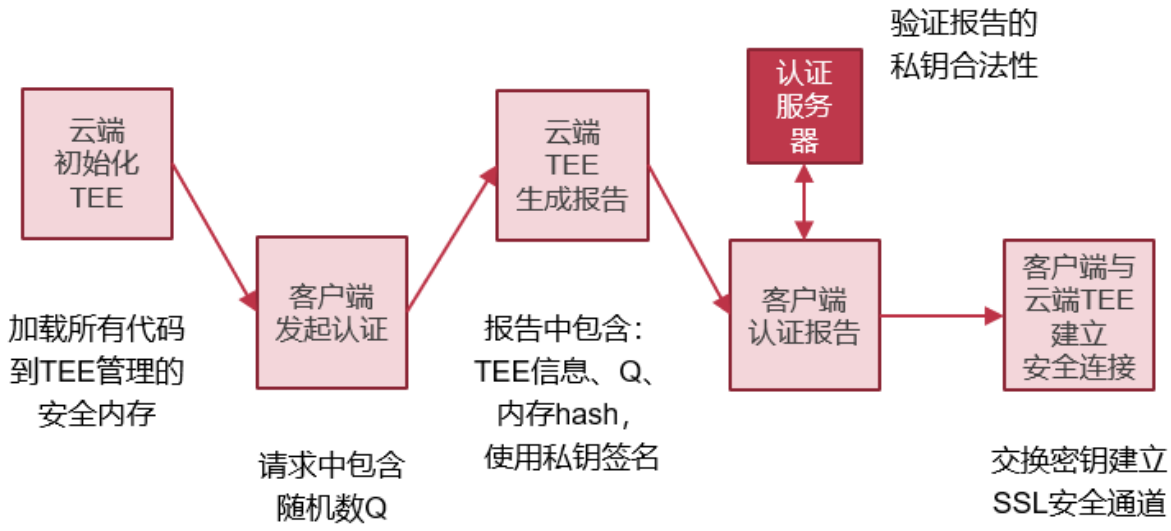
- **启动后的度量（动态度量）**
  - 如Intel TXT（Trusted eXecution Technology）
  - 支持启动后对区域内存进行认证，如认证Hypervisor

TEE的初始化与远程证明

1. **TEE初始化（TEE服务商）**
   - 初始化硬件环境，并加载软件进入TEE
2. **远程证明（三方交互）**
   - 用户向TEE发起一个认证请求
   - TEE生成认证，签名后返回给用户
   - 用户向验证服务器确认认证有效性
3. **建立安全信道（用户与TEE）**
   - 用户与TEE交换密钥后建立加密通道
4. **通过信道输入数据，并调用算法执行（用户与算法）**

TEE

远程认证(Remote Attention)



远程证明的安全边界：何时无效?

- **远程证明可能被攻击成功的场景**
  - TEE内部的私钥被泄露或被伪造
  - 认证服务器签名私钥被泄露或被伪造
  - 加密算法被攻破（如量子计算等）
  - 依然基于传统的PKI体系（如CA等）
- **信任方**
  - TEE硬件制造商、远程证明服务提供方

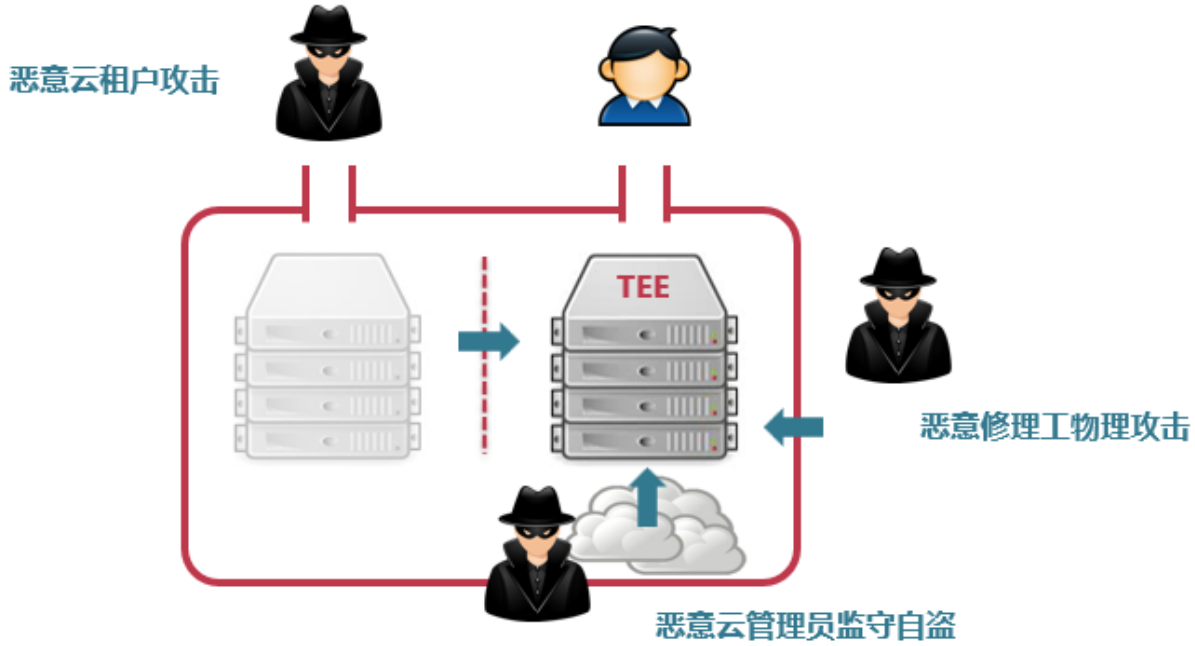
4. TEE技术点二：隔离执行

云计算的安全等级：从技术的角度

安全等级	篡改内容	窃取数据	观察活动
最高	不允许	不允许	不允许
高	不允许	不允许	允许
中	不允许	允许	允许
低（现状）	允许	允许	允许

- **篡改内容**：管理员可修改算法或数据使计算得出不正确的结果
- **窃取数据**：管理员可窃取算法执行过程中在内存或存储中的机密数据
- **观察活动**：管理员可观察到算法具体执行过程中与内存或存储的交互活动
  - 如：访存行为、磁盘I/O行为、网络I/O行为，等等

TEE的攻击面



攻击-1：恶意特权软件的攻击

- **特权软件：如操作系统、Hypervisor**
  - 通常直接控制所有硬件资源
  - 为应用程序提供抽象，如系统调用
- **来自恶意特权软件的攻击 —— 降维打击**
  - 可直接窃取或篡改用户数据和代码（隐私性与完整性攻击）
    - 包括CPU状态、内存、存储、网络等
  - 可拒绝为应用提供服务（可用性攻击）

防御-1：基于访问控制隔离特权软件

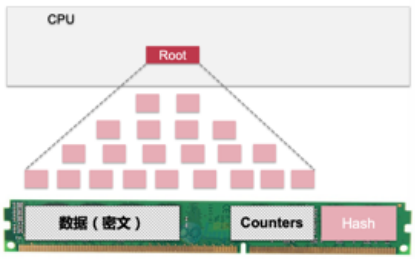
- **利用硬件隔离机制限制特权软件的权限**
  - 方法-1：直接新增硬件隔离能力保护TEE范围的硬件
  - 方法-2：引入更底层的特权软件（属于可信计算基）
- **典型实现**
  - 页表机制，主流CPU均支持
  - PMP（Physical Memory Protection），如：RISC-V
  - PRM（Processor Reserved Memory），如：Intel SGX
  - 嵌套虚拟化，如：CloudVisor

攻击-2：恶意硬件的攻击

- **硬件攻击的特点**
  - 攻击难度更高：通常需要直接接触硬件（如伪装成维修人员）
  - 防御难度更高：直接绕过软件的防御（如访问权限）
- **典型的硬件攻击**
  - 内存中间人攻击（威胁机密性、完整性与可用性）
  - 系统总线嗅探攻击（威胁机密性）
  - 非易失性内存窃取（威胁机密性、完整性）
  - 恶意DMA攻击（威胁机密性、完整性）
  - 内存冷冻启动攻击（威胁机密性）

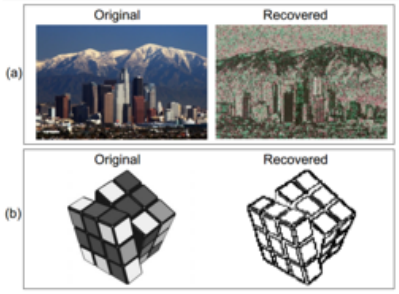
防御-2：基于内存加密防御物理攻击

- **隐私性保护：内存加密**
  - 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，进行加密和解密操作
- **完整性保护：Hash Tree**
  - 对内存数据计算一级hash，对一级hash计算二级hash，形成树
  - CPU内部仅保存root hash
  - 当内存中的数据被修改时，更新Merkle Tree



攻击-3：基于访问模式的侧信道攻击

- **侧信道（隐秘信道）**
  - 原本无法直接通信的两方，通过原本不被用于通信的机制进行数据传输
  - 常见的隐秘信道：时间、功耗、电磁泄露、声音等
- **TEE内代码执行过程会暴露访问模式**
  - 磁盘和网络访问模式对特权软件可见
  - 内存访问模式可通过缓存和页表泄露
  - 例-1，利用页表攻击Intel SGX窃取TEE内数据
  - 例-2，利用缓存攻击TrustZone窃取TEE数据



防御-3：混淆访问模式+减少资源共享

- **混淆访问模式（通常在应用层）**
  - 使用常量时间算法以消除时间侧信道
  - 使用ORAM（Oblivious RAM）以混淆访存模式
    - 使用固定模式访问资源以消除信息泄露
- **减少资源共享（通常在系统层）**
  - 空间隔离：为TEE使用单独的CPU核、内存和外设
  - 时间隔离：在切换时刷掉所有共享状态
    - 如缓存、TLB等可被观测到的状态

```
/* 传统实现方式 */
if (secret == 0)
    x = a + b;
else
    x = a / b;

/* 常量时间实现方式 */
v1 = a + b;
v2 = a / b;
cond = (secret == 0)
x = cmov(cond, v1, v2)
```