



# TEE的国内外发展现状和趋势

问题，挑战，解决方案

上海交通大学·夏虞斌

2015-5-22



- 上海 **瓶钵信息科技®** 有限公司，依托于上海交通大学软件学院、并行与分布式系统研究所成立，公司成员均为来自上海交通大学和复旦大学的博士和硕士
- 公司重点关注**系统安全**，具有超强的研发能力，专注于业界前沿，致力于将研究所的研究成果转化为产品



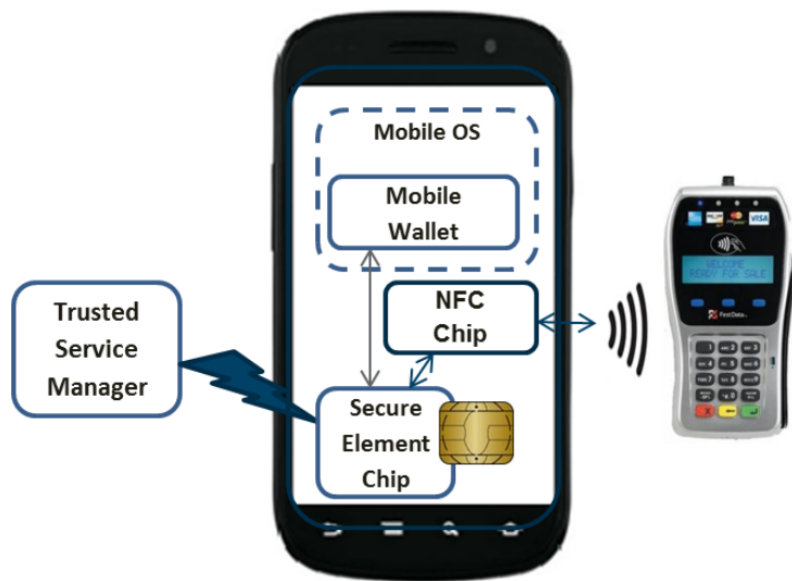
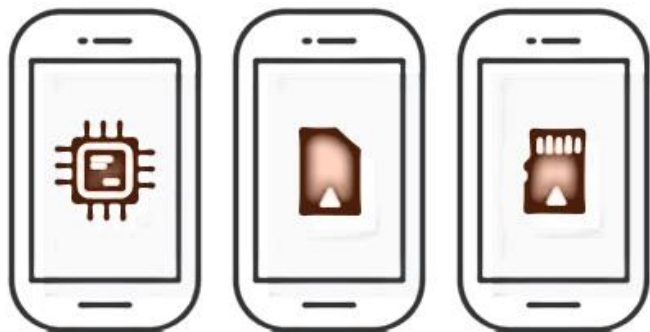
安全性与可部署性的权衡

# SE和HCE的安全性对比

# 基于SE的移动支付方案



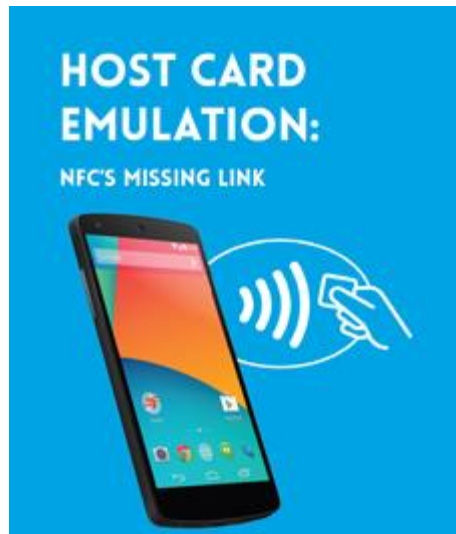
- Secure Element
  - 能提供安全存储、安全执行的硬件
  - 例如：安全SIM卡、安全存储卡等



# 基于HCE的移动支付方案

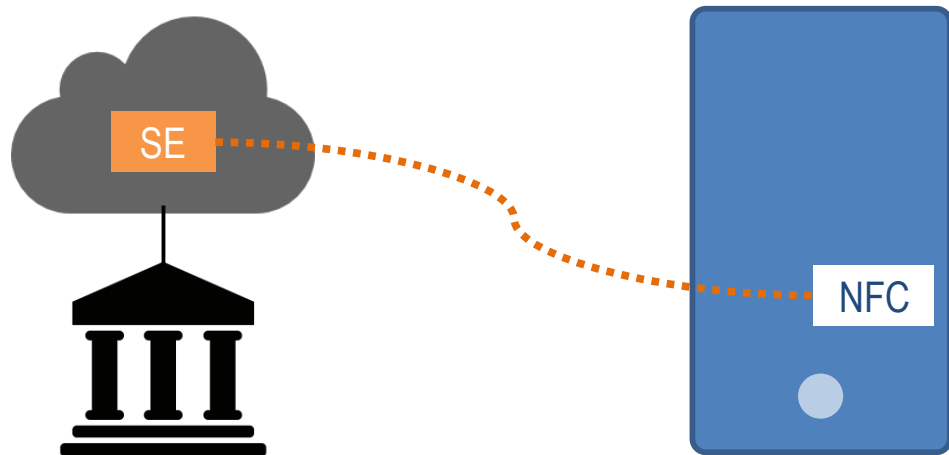


- HCE : Host Card Emulation
  - Google于2013年推出
- 兼容性强，易部署
  - 兼容普通SIM卡，不依赖于SE和TSM
  - 仅需手机支持NFC功能





- 现有HCE技术的安全缺陷
  - 基于本地（卡信息 | Token）：安全性差，受恶意软件的威胁
  - 基于云端（CloudSE）：支付时手机需要连网，体验差



# 为什么HCE不安全？



- 将关键数据保存在本地
  - 需要相信整个Android操作系统
  - 一旦手机被Root，攻击者可获得任意数据
- 为什么SE安全？
  - 和支付相关的数据和执行均在SE内部完成
  - SE与Android完全独立，能够抵御物理攻击
  - 即使手机被root也不受影响



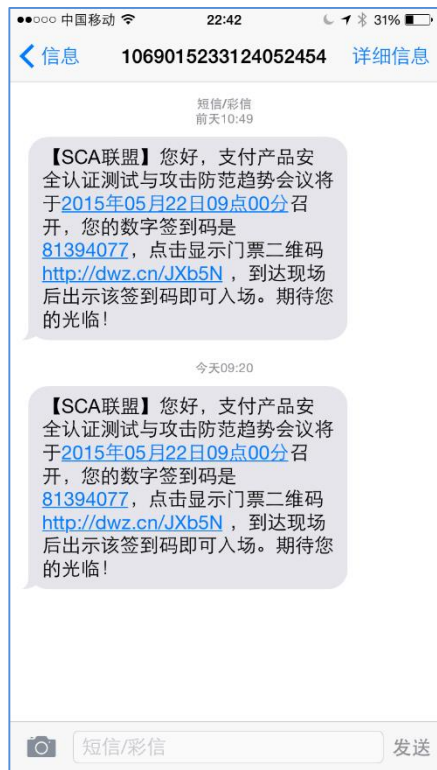
- 利用系统漏洞获得手机的所有权限
  - Android设备：Root；iOS设备：越狱
  - 可运行任意代码，窃取或篡改内存和存储中的任意数据
- Root有多容易？
  - 恶意软件，欺骗用户安装使用（如：一键root）
  - 利用浏览器漏洞，诱使用户点击链接或扫描二维码



# Root方式的多样性



## iOS 4.3.3 越狱：访问网页即可



# 如何提高HCE的安全？



- 将关键数据保存在云端（即CloudSE）



- 支付时需要手机连网

- 将关键数据保存在手机的SE内



- 为什么不直接用SE？

- 结论1：HCE依然存在安全问题

- 问题：能否做到用手机不信手机？

使用手机的硬件：保证兼容性

不信手机的软件：防Root攻击

安全硬件与可信执行环境



# TRUSTZONE & TEE

# ARM TrustZone 技术



TrustZone：在CPU上同时运行Android和一个隔离的可信操作系统

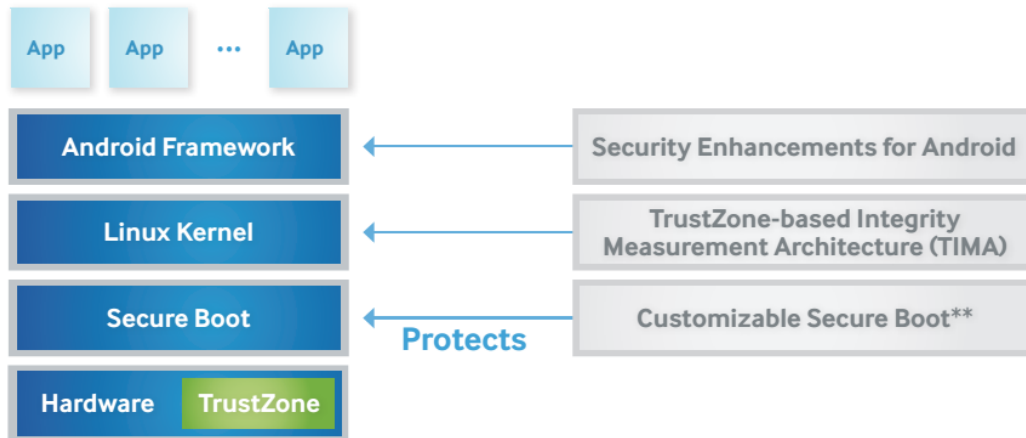


# TrustZone应用实例1：三星TIMA



- 动态内核完整性保护

- TIMA: TrustZone-based Integrity Measurement Architecture
- 检测内核代码段和关键数据的完整性



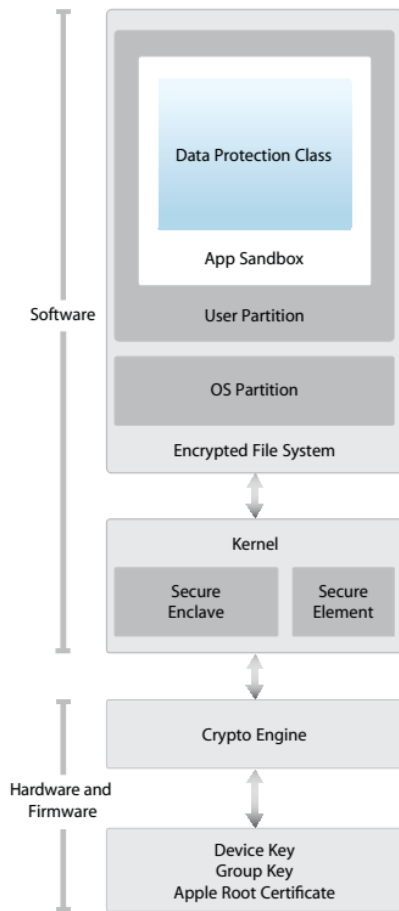
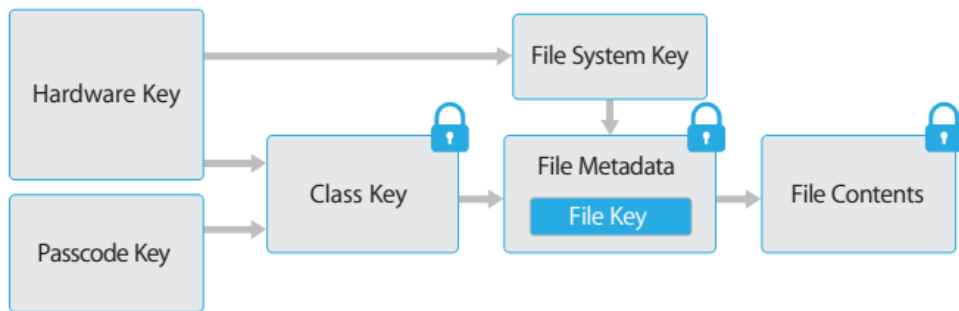
\* 资料引自三星

# TrustZone应用实例2：iOS Enclave



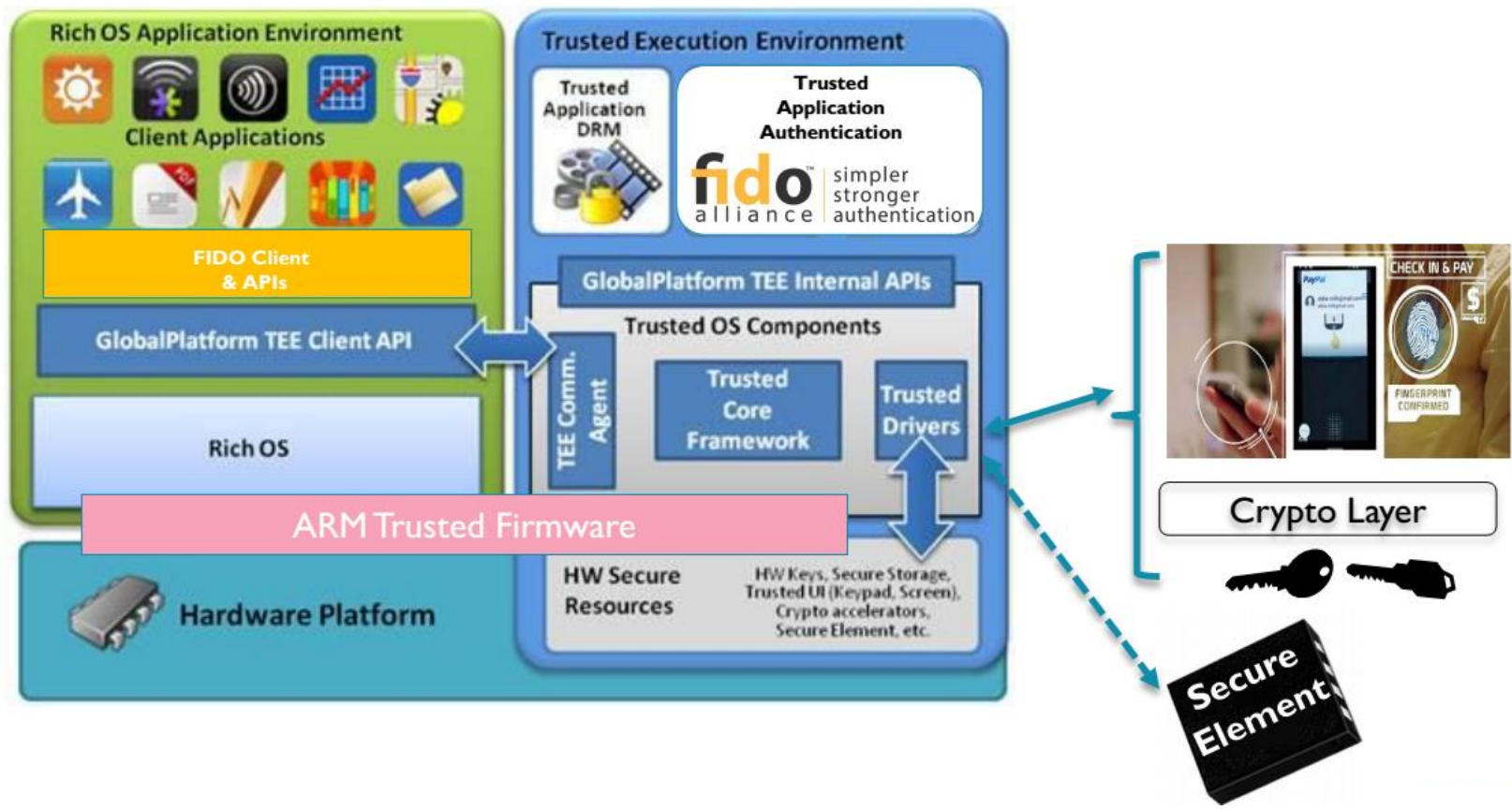
- iPhone 5s 指纹识别的基础

- 基于TrustZone等技术实现Enclave与外界强隔离
- 所有指纹相关数据均通过Enclave保存和访问





\* 资料引自Apple

# TrustZone应用实例3：与FIDO的结合



# TEE与移动支付相结合



- 1、使用手机的硬件 
  - TrustZone将处理器一分为二
- 2、不信手机的软件 
  - 提供与Android完全隔离的执行环境
  - 即使Root也无法破坏隔离性
- 结论2：TEE可做到“用手机但不信手机”
  - 问题：HCE+TEE 能否达到与SE同等的安全性？





## TEE的安全性

# TEE+HCE的三个安全问题



- 1、TEE的存储安全性不够
- 2、TEE防御物理攻击能力较弱
- 3、TEE实现可能存在漏洞

# 1、TEE的存储安全性不够



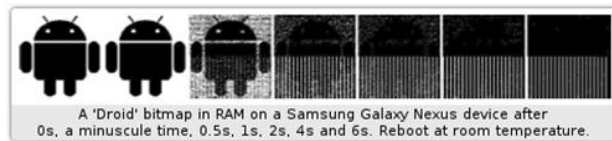
- HCE中的关键数据
  - 卡信息和Token信息
  - 一旦被窃则会直接导致用户经济损失
- 为何不加密保存数据？
  - 密钥本身依然需要安全存储
- SE：将数据保存在芯片内部
  - 通过物理手段保证数据不会流出芯片



## 2、TEE无法防御物理攻击



- 常见的物理攻击
  - 直接扫描物理内存获取明文
  - 利用内存数据残留的冷启动攻击
  - 显微镜读取芯片内部数据
- SE如何防御物理攻击？
  - 所有数据保存在芯片内部
  - 检测到物理攻击则立刻自毁



Muller等研究者在三星Galaxy Nexus手机通过FROST攻击获得硬盘加密密钥

# 3、TEE的实现存在漏洞



- 系统复杂度高
  - TEE是一个完整的操作系统
  - 可扩展性：需要支持不同可信应用的运行
  - 可交互性：需要与Android环境进行交互
- 生态尚未稳定
  - 许多TEE系统同时存在
  - 系统级漏洞：内外交互、设备管理等

# 2013 : Moto手机的TrustZone内核被破解



- 2013年，Moto部分手机TrustZone内核被破解
- 可解锁Bootloader，加载任意系统

## Unlocking the Motorola Bootloader

posted by Dan Rosenberg @ 4/08/2013 11:29:00 AM

I recently spent some time dissecting the bootloader used on Motorola's latest Android devices, the Atrix HD, Razr HD, and Razr M. The consumer editions of these devices ship with a locked bootloader, which prevents booting kernel and system images not signed by Motorola or a carrier. In this blog post, I will present my findings, which include details of how to exploit a vulnerability in the Motorola **TrustZone** kernel to permanently unlock the bootloaders on these phones.

These three devices are the first Motorola Android phones to utilize the Qualcomm MSM8960 chipset, a break from a long tradition of OMAP-based Motorola devices. Additionally, these three devices were released in both "consumer" and "developer" editions. The developer editions of these models support bootloader unlocking, allowing the user to voluntarily void the manufacturer warranty to allow installation of custom kernels and system images not signed by authorized parties. However, the consumer editions ship with a locked bootloader, preventing these types of modifications.

来源: <http://blog.azimuthsecurity.com/2013/04/unlocking-motorola-bootloader.html>



- 高通漏洞 : CVE-2014-4322
  - drivers/misc/qseecom.c文件存在漏洞
  - 没有判断offset和length的正确范围
  - 可能导致攻击代码提权或DoS攻击

来源 : <https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2014-4322>



- 没有检查传入指针参数的范围
  - g\_fs\_status为0, 可在s+16的位置写入0
  - 00 00 = MOV r0, r0 | 00 00 00 00 = ANDEQ r0, r0, r0
  - 攻击方式: 删除任意安全检查代码

```
int __tzbsp_oem_discretix(struct_p * s, size_t len) {
    if (len != 0x14) {
        return -16;
    }
    s->status = g_fs_status; // *(int *)(s + 16) = g_fs_status
    ...
}
```





- S5指纹识别系统
  - 使用TrustZone来保存指纹数据
  - 对数据进行加密
- 系统漏洞 ( 2015年4月 )
  - 攻击者root内核后, 可直接访问指纹设备获取指纹
  - 攻击者通过指纹冒充用户



来源 : <http://securityaffairs.co/wordpress/36326/hacking/samsung-s5-flaw-steal-fingerprints.html>



- 来自Android：交互接口、TEE自身驱动
- 来自硬件：CPU、内存、设备总线、传感器
- 来自启动：eMMC的安全启动
- 来自内部：安全应用的Bug



- TEE的存储安全性不够
    - 无法保证关键数据（如Token）的安全
  - TEE防御物理攻击能力较差
    - 在外部内存中的数据 and 代码可能被篡改
  - TEE实现可能存在漏洞
    - 代码越来越多导致bug数量增多
    - 多种TEE系统并存，处于快速发展阶段
- SE：数据不在内存中
- SE：能抵御物理攻击
- SE：简单，bug少
- **结论3：当前TEE依然无法达到SE的安全等级**

增强TEE的安全——我们的努力

 TRUSTKERNEL的产品：T6



# T6

- 更安全的数据存储
- 更强的物理攻击防御
- 更小的可信基

# T6 : TrustZone TEE 与 Secure OS

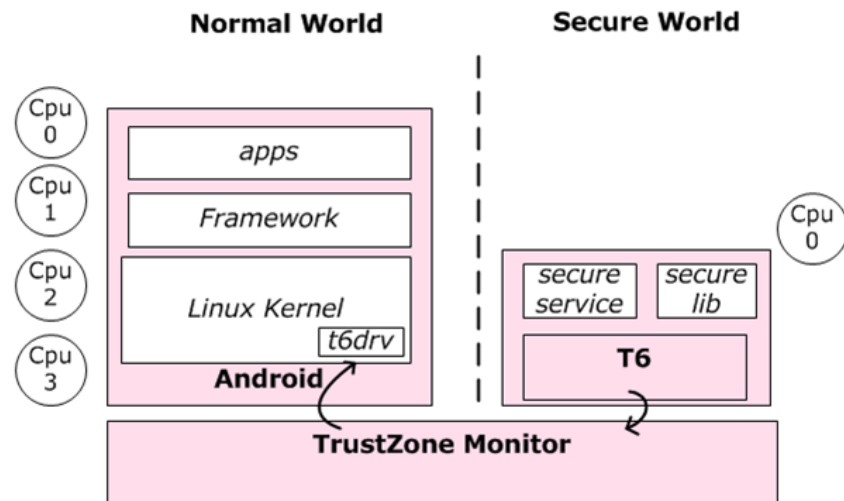


- **高安全：动静态保护**

- SecureBoot完整信任链
- Trusted App (TA)签名验证与保护
- TA中不同域与库间互相隔离

- **小尺寸：轻量而易用**

- 核心安全代码仅 6,000 行
- 内存占用量极少
- 开发库支持完善
  - 加密，安全界面，文件系统，libc，网络等
- 兼容GlobalPlatform API





- On-chip TEE
  - 所有明文数据均只在SoC内部，外部内存中只保存密文数据
  - 全系统保护，支持包括TA在内的TEE整体
  - 支持后向兼容，保持对现有TA透明，对TEE和TA内存大小没有限制
  - 基于PUF的密钥管理，增强存储安全
  - 有效保证Token等关键数据的安全
- 威胁模型
  - 仅SoC可信：对SoC本身进行物理攻击的难度极高
  - 能够应对目前常见的物理攻击

# HCE与T6的结合



	易部署	离线支付	安全存储	防物理攻击	可信基小
SE		✓	✓	✓	✓
HCE-CloudSE	✓		✓	✓	
HCE-Token	✓	✓			
HCE-TEE	✓	✓			
HCE-T6	✓	✓	✓	✓	✓

结论4：通过安全的系统设计，TEE可以接近SE的安全等级





- 结论1：HCE依然存在安全问题
- 结论2：TEE可做到“用手机但不信手机”
- 结论3：TEE当前依然无法达到SE的安全等级
- 结论4：通过安全的系统设计，TEE可以接近SE的安全等级



## TEE及其相关应用



- Secure hardware tokens
- Mobile payment
- BYOD
- Runtime integrity verification
- Trusted user interface
- Remote enablement/disablement
- Automotive (trust vs. safety)
- Secure isolation, Remote attestation
- DRM, HDCP, secure NFC in P2P mode
- Any other operation that requires verifiable

# 场景1：支付中的PIN码安全输入



**Merchant Site**  
Merchant World

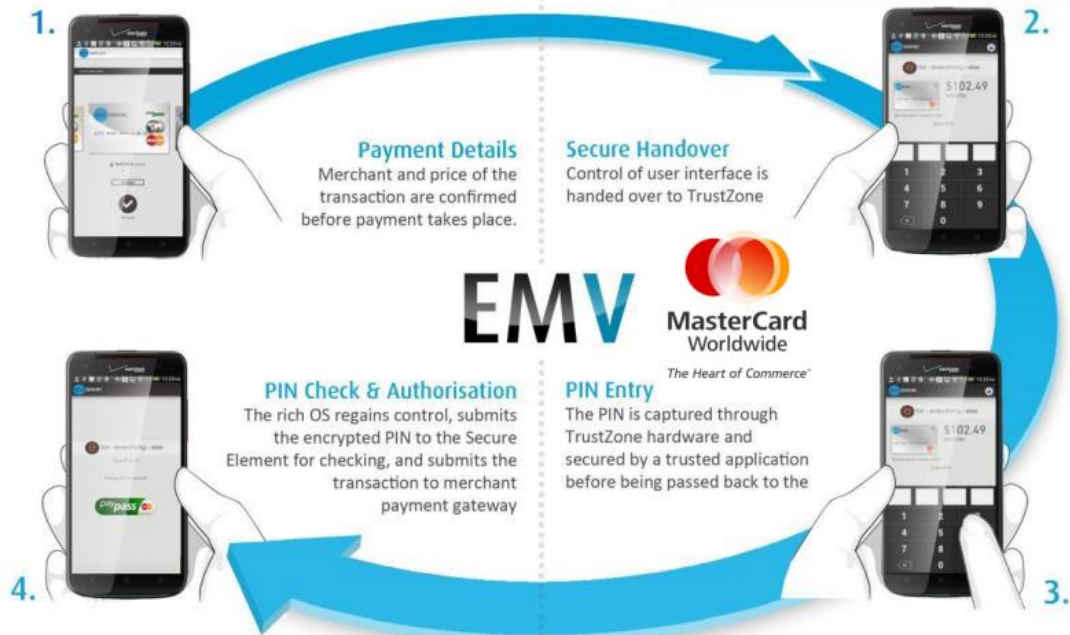


**Merchant Shopping**  
The merchant 'pay now' element drives the TrustZone app to launch, delivering the necessary information to be displayed



**Merchant Confirm**  
A confirmation is returned to the merchant to confirm the payment was successfully taken

## Rich OS Normal World

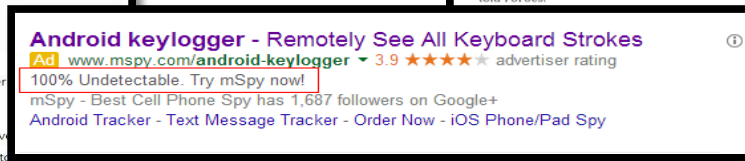
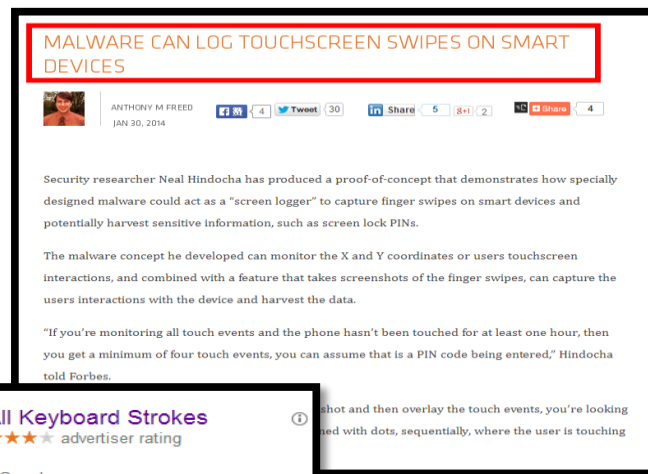


\* 资料引自ARM

# TrustZone应用实例：可信UI



- 针对用户输入的恶意软件层出不穷
  - 在root环境下KeyLogger可截获所有输入（密码、信用卡号等），并绕过所有已知的防御方法



# TrustZone应用实例：可信UI



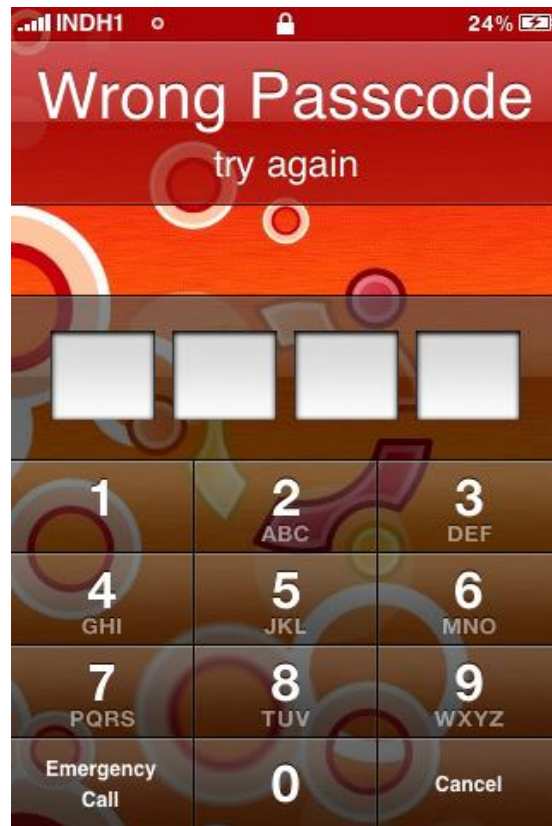
- 用户**输入的信息**可能被窃取、篡改
  - 记录下用户的支付密码
  - 修改用户的输入，如将支付1元改为100元
- 用户**看到的信息**可能被窃取、篡改
  - 如显示收款人为A，实际为B
  - 钓鱼攻击欺骗用户输入密码
  - 通过截屏获取用户输入的密码



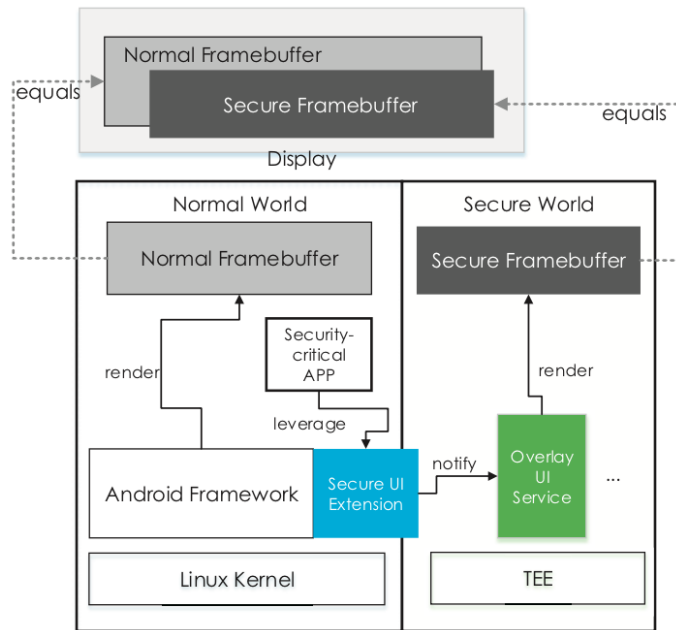
# 侧信道攻击窃取PIN码



- 攻击方式
  - 通过陀螺仪获取晃动信息
  - 通过晃动信息判断点击位置
  - 通过点击位置判断输入PIN码
  - 通过机器学习提高准确率
    - 准确率高达90%
- 攻击前提
  - 恶意程序获得访问陀螺仪的权限
  - 属于系统攻击：SE同样存在该问题



# OverlayUI方案：芯片级安全UI控件



- 安全输入与显示控件，阻止KeyLogger窃取和钓鱼攻击
- 与现有系统的无缝整合，用户零学习成本
- 控制传感器，防止侧信道攻击



# 场景2：使用TEE保护REE的关键数据



- 应用对关键数据的保护不够
  - 大量应用在内存和数据库中明文存放关键数据
  - 如登录密码、银行账号、信用卡号等
- 数据暴露原因
  - 数据非安全删除
  - OS的数据缓冲区
  - App的数据缓存机制
  - 内存泄露

通过对内存dump的扫描发现：在14个Android流行应用中，13个会在内存和磁盘中保存明文的关键数据；9个会在内存中一直保存明文！

# 应用关键数据暴露在不可信内存中



App	Extracted Cleartext Data
Email	password, email contents, subjects, from/to, contacts
OI Notepad (doc)	document and metadata
KeePass (password mgr)	app password, all stored passwords & descriptions
Pageonce (finance)	password, transactions, bank account information
Facebook (social)	wall posts and messages

(a) Examples of data extracted from RAM / DB.

App	Data	When App Uses Data
Email	password	user/automatic refresh
	subjects	on the email list screen
	contents	user opens the email
OI Notepad	note title	on the note list screen
	note body	user edits the note
KeePass	master password	app launches
	entry name	on the entry list screen
	entry password	user opens the entry

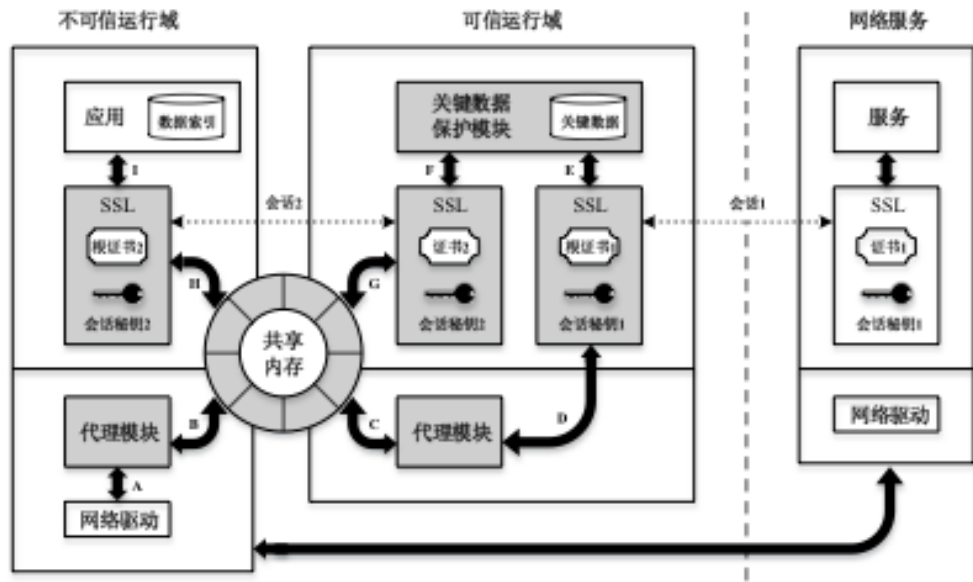
(c) Example usage of hoarded data by apps.

App	Description	Extracted Cleartext Data			Cleartext Data Hoarding					
		Pass-word	Cont-ents	Meta-data	RAM			SQLite DB		
					Pass-word	Cont-ents	Meta-data	Pass-word	Cont-ents	Meta-data
Email	email (default)	Y	Y	Y	Y		Y	Y	Y	Y
GMail	email		Y	Y					Y	Y
Y! Mail	email	Y	Y	Y					Y	Y
GDocs	documents			Y			Y		Y	Y
OI Notepad	documents		Y	Y		Y	Y			
DropBox	documents			Y			Y			Y
KeePass	password mgr	Y	Y	Y	Y	Y	Y			
Keeper	password mgr	Y	Y	Y	Y		Y			
Amazon	commerce									
Pageonce	finance	Y	Y	Y	Y	Y	Y			Y
Mint	finance		Y	Y		Y	Y		Y	Y
Google+	social		Y	Y					Y	Y
Facebook	social		Y	Y					Y	Y
LinkedIn	social		Y	Y			Y		Y	Y

(b) Exposure of cleartext sensitive data across all 14 apps.

数据来源: Tang, Yang, et al. "CleanOS: Limiting Mobile Data Exposure with Idle Eviction." *OSDI*. Vol. 12. 2012.

# 我们的方案: Android敏感数据保护



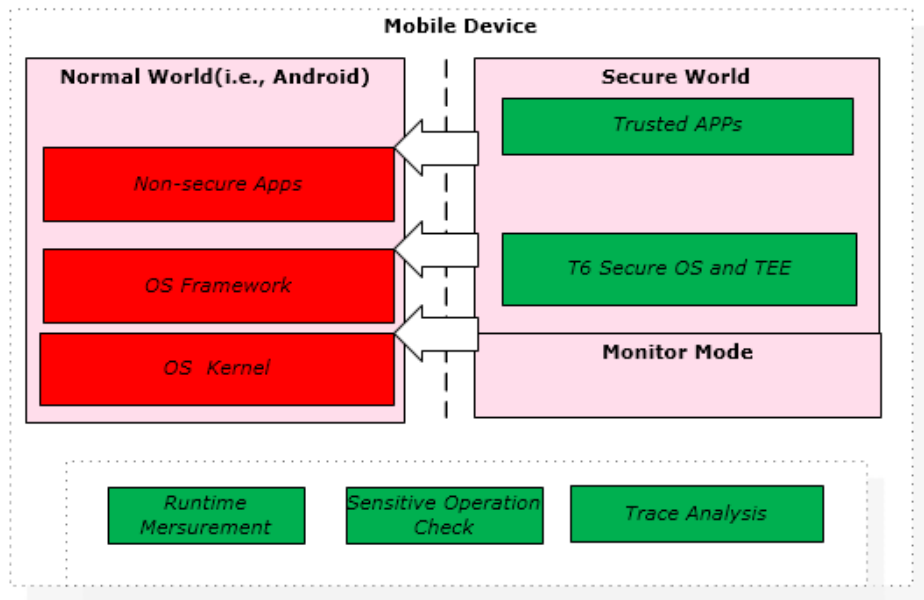
- 关键数据保存在可信域，不出现在普通域，防御恶意软件窃取
- 仅在可信域访问关键数据（如网络发送），保证可靠的访问策略

# 场景3：使用TEE加固REE安全



- REE安全问题严重
  - 如何实时对REE进行攻击防护
- 市场已有方案：
  - TIMA: TrustZone-based Integrity Measurement Architecture
  - 采用定期检测方法

# RTFence方案: Android内核安全增强



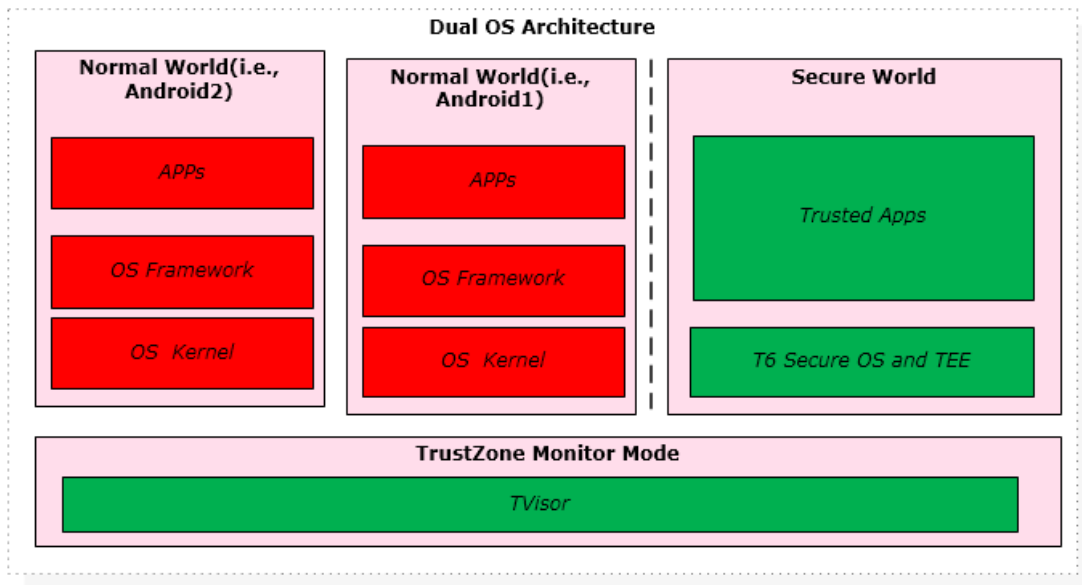
- TEE中集成内核关键功能，实时监控内核，保证完整性
- 有效预防Rootkit攻击

# 场景4：基于TrustZone的多操作系统



- 一部手机如何同时运行两个不同的系统
  - 在没有硬件虚拟化支持的前提下
  - 例如：一个Android系统，一个传统的功能机系统
- 隔离安全
  - 如何保证两个系统互不影响
  - 如何在两个系统中快速切换

# Tvisor方案: 轻量级ARM Hypervisor



- 双系统支持，毫秒级切换，支持现有Android系统
- 内核级隔离，阻止Rootkit扩散

# 场景5：TEE的多样性



- 不同客户对TEE系统的需求不同
- 如何在同一平台支持不同的TEE？



# 解决方案：vTZ



- 硬件保证隔离，安全性较高
- 仅通过SMC指令进行切换
- 双系统运行环境



- 软件保证隔离，安全性相对较低
- 陷入虚拟机监控器的方式更多更灵活
- 多系统运行环境

# 虚拟化与TrustZone的融合



- 为每个虚拟机虚拟一个或多个secure world
- 使用TrustZone保证虚拟secure world安全



## T6

### TEE平台与Secure OS

- 自主开发支持TrustZone的安全操作系统与开发平台
- 符合Global Platform标准
- 支持安全应用：DRM、安全支付、密码管理等
- 支持灵活定制和二次开发

## TVisor

### 轻量级ARM Hypervisor

- 高安全双操作系统方案
- 支持多样的操作系统
- 无需硬件虚拟化支持

## TrustVDI

### BYOD定制安全瘦客户系统

- 从客户端到云端的BYOD瘦客户安全防护方案
- 数据访问限制在安全域内
- 保证端到端的访问策略

## OverlayUI

### 芯片级安全UI控件

- 系统级安全控件
- 应用程序可无缝使用
- 防御Android程序钓鱼攻击

## RTFence

### Android内核安全增强

- 运行时内核事实安全监控
- 可防范Rootkit等攻击

## TZProxy

### Android敏感数据保护

- 密码管理与保护
- 敏感数据无缝替换

## T6-m

### 高安全级T6系统

- 运行在SoC芯片内部
- 不占外部内存，抵御物理攻击



- TEE的应用
  - TEE可以为移动支付提供更安全的存储和执行环境
  - TEE可从系统层面为移动支付提供更广的安全
- TEE的机遇和挑战
  - TEE本身会面临越来越多的安全挑战
  - 虚拟化等新的硬件特性会给TEE带来更多机会
- 瓶钵科技，关注TEE的应用和安全



感谢您的关注，期待与您合作

<http://www.trustkernel.com>

瓶钵科技