看雪

SDC·2019

2019 安全开发者峰会

Security
Development
Conference

IOT中的SE芯片安全

潘少华　　江苏知道创宇

目录

1. IOT设备安全SE情况

2. SE芯片具有的特性和风险

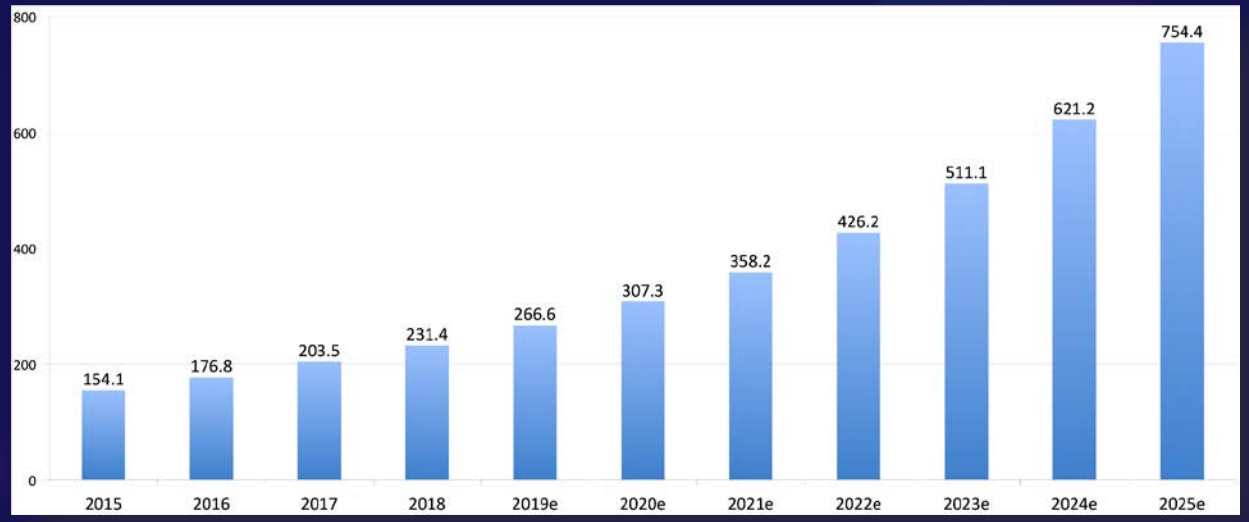
3. SE芯片安全分析及防护

IOT设备安全情况

预计到2025年全球物联网设备部署将超过750亿台

据数据调查统计公司Statista统计，2018年全球物联网部署设备数量已达到231.4亿台，根据预测2019年将达到266亿台，预计到2025年，物联网部署设备数将超过750亿台。

随着我国“中国制造2025”战略推进、两化深度融合、以及多层面互联互通政策带来的产业大规模提档升级，物联网应用在国内各行业也得到了越来越多的部署。



数据来源：Statista @ 2019

暴露在互联网上的物联网设备超过6000万台

我们利用国内网络空间探测引擎 ZoomEye 在2018年对物联网设备进行了全年监测发现，全球暴露在互联网上的物联网设备已超过6000万台：

路由器 2452万台
华为 809万台

网络摄像头 1253万台
海康威视 629万台

NAS 319万台
QNAP 159万台

打印机 68万台
Brother 7万台

数据来源：Zoomeye.org @ 2019

接入互联网的物联网设备所面临的安全威胁日益彰显

路由器、摄像头、NAS、打印机这四类是目前监测到暴露数量最多且最易受到安全威胁的物联网设备，仅从这些设备分析他们面临着非常严重的安全威胁。

全球路由器有将近400万个Telnet服务暴露在互联网上，一旦攻击者通过telnet服务登录到路由器上，意味着接入内部局域网控制如物联网网关、摄像头等设备，威胁人们的隐私、财产和生命安全。

摄像头暴露的HTTP服务数量最多，假设暴露在互联网中的摄像头有10%存在弱口令，他们极易变成僵尸网络的受控机，那么将有可能制造出高达Tbps级别的DDoS攻击。

攻击者可以连接到存在漏洞的NAS设备上编写简单的持久性shell，进而执行更多的命令。有的执行命令可以允许他们转储NAS设备的完整数据库，其中包含电子邮件、用户名以及MD5校验值的用户隐私数据。

全球仅有不到2%的打印机是真正安全的。有相当一部分打印机的HTTP服务没有启用必要的登录认证机制，导致远程用户不需要登录即可访问，会让设备上的文档信息处在数据泄露的高风险威胁中。

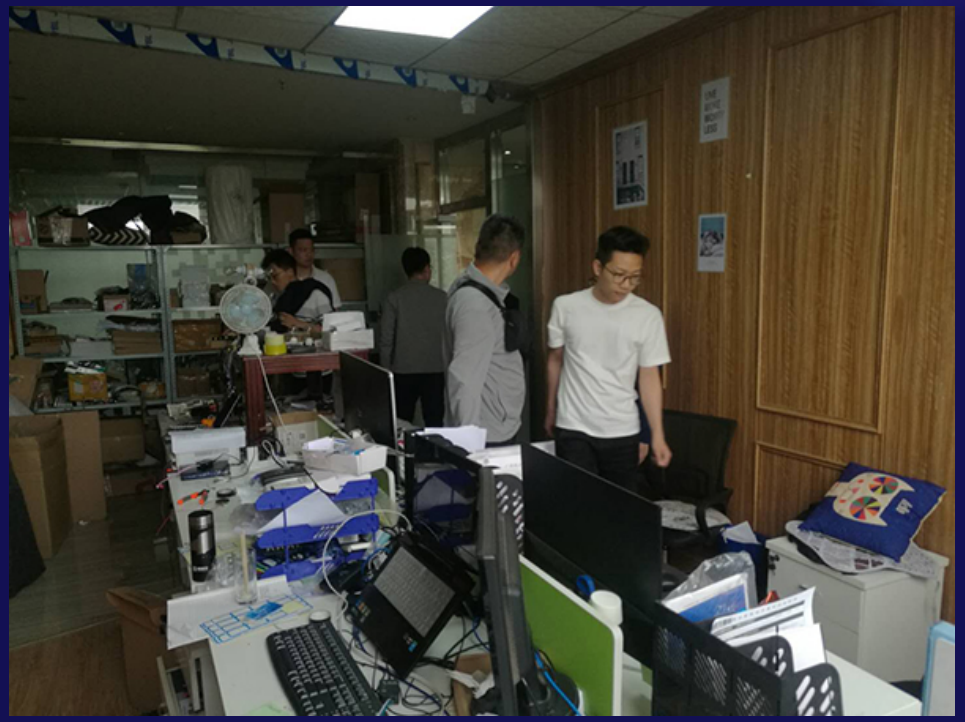
物联网安全风险——媒体报导

央视报导智能家居遭黑客入侵

- 2019年1月，CCTV4报道了智能家居遭黑客入侵事件，包括智能门铃、智能电视、扫地机器人等等被入侵后监视用户的安全风险。
- 包括智能锁在内的智能家居面临着黑客入侵风险，
- 对用户隐私安全、财产安全甚至人身安全造成了威胁。

物联网安全风险——媒体报导

湖北首例入侵物联网系统案告破，10万设备离线



2019年3月21日至22日，位于光谷总部国际的“微锋”（化名）科技有限公司的多台物联网终端设备出现故障：自助洗衣机、摇摇车、抓娃娃机等均脱网无法正常运行。经统计，共100余台设备被恶意升级无法使用、10万台设备离线，造成了重大经济损失。

嫌疑人谢某系“微锋”公司前员工，离职后与王某共同成立了“微天地”科技公司，成为“微锋”竞争对手。谢某、王某破解了“微锋”公司的物联网服务器，利用系统漏洞将终端设备恶意升级，导致100余台设备系统损坏，无法正常工作；同时模拟终端设备，以每秒3至4千条的速度给服务器发送伪造离线报文，导致10万台设备离线。

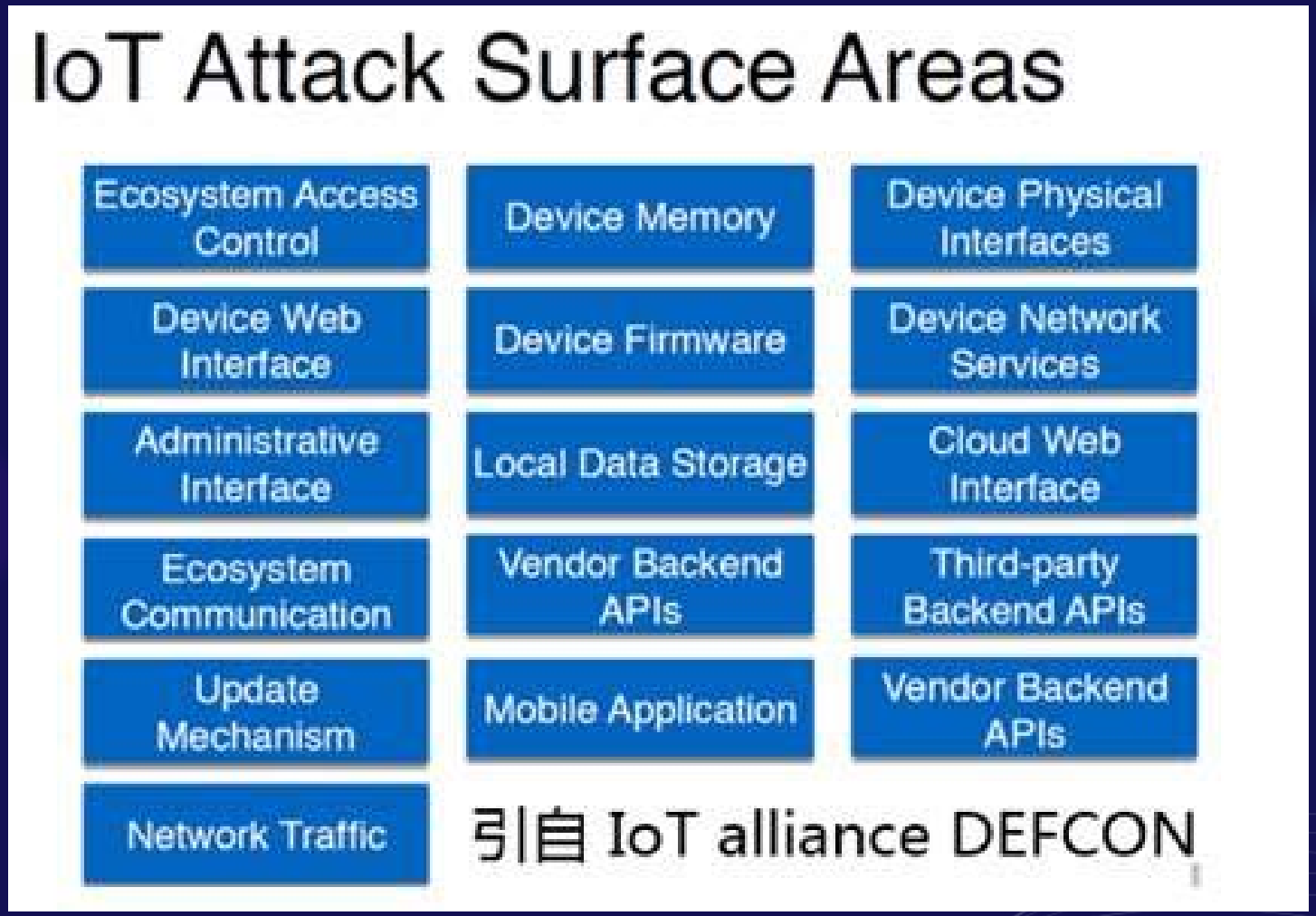
物联网安全威胁


目标用户
①目标用户登录站点
②站点给予目标用户一个
Session ID
③攻击者使用某种方式获取目标用户的
Session ID
④攻击者获得合法会话
攻击者

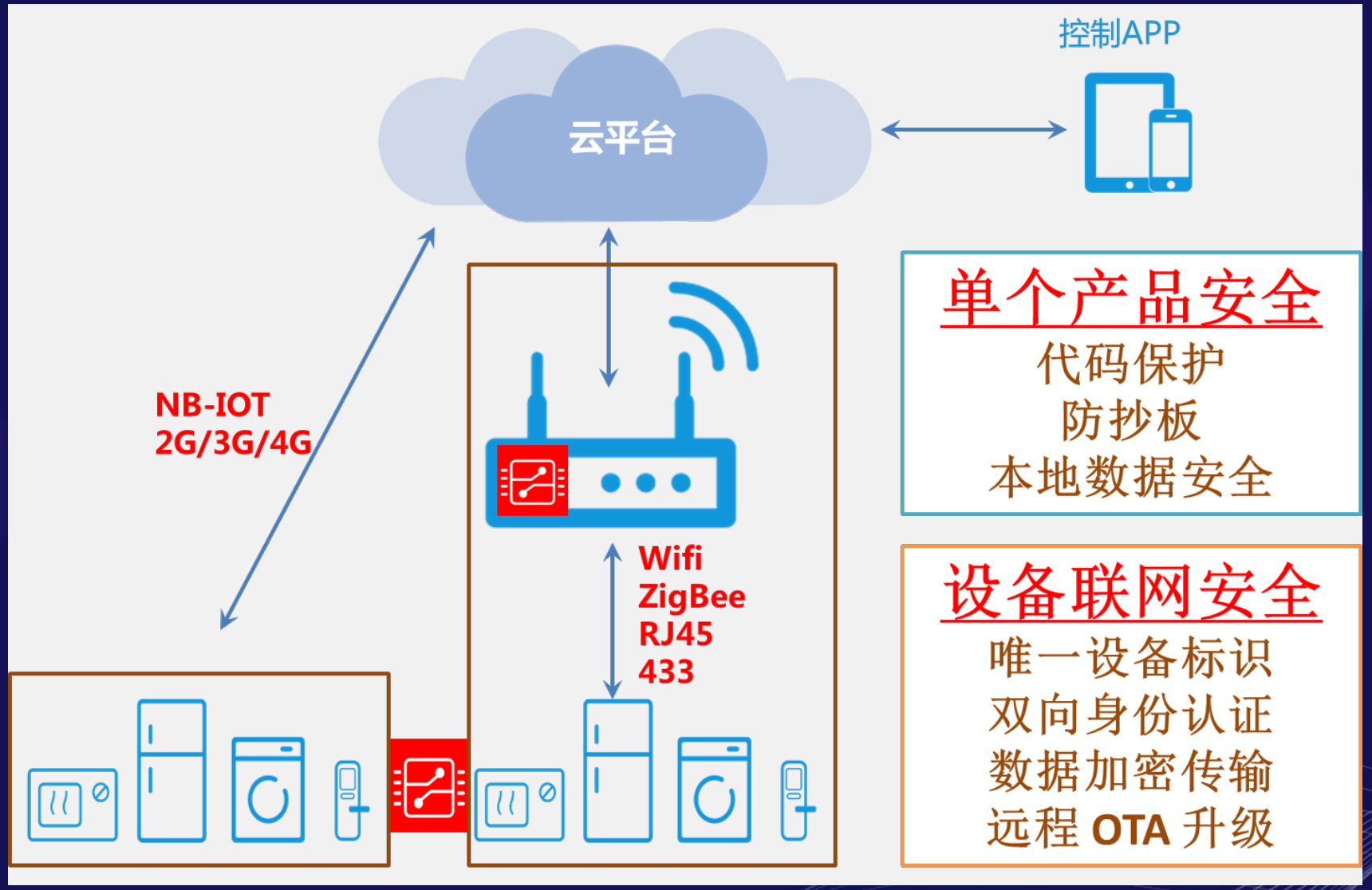

中间人攻击
Man-in-the-middle attack
Original connection
New connection
Man-in-the middle, Phisher,
or annonymous proxy

DDOS

物联网安全威胁


IoT Attack Surface Areas
Ecosystem Access Control
Device Memory
Device Physical Interfaces
Device Web Interface
Device Firmware
Device Network Services
Administrative Interface
Local Data Storage
Cloud Web Interface
Ecosystem Communication
Vendor Backend APIs
Third-party Backend APIs
Update Mechanism
Mobile Application
Vendor Backend APIs
Network Traffic
引自 IoT alliance DEFCON

物联网设备安全需求



单个产品安全

代码保护
防抄板
本地数据安全

设备联网安全

唯一设备标识
双向身份认证
数据加密传输
远程 OTA 升级

物联网安全——硬件安全

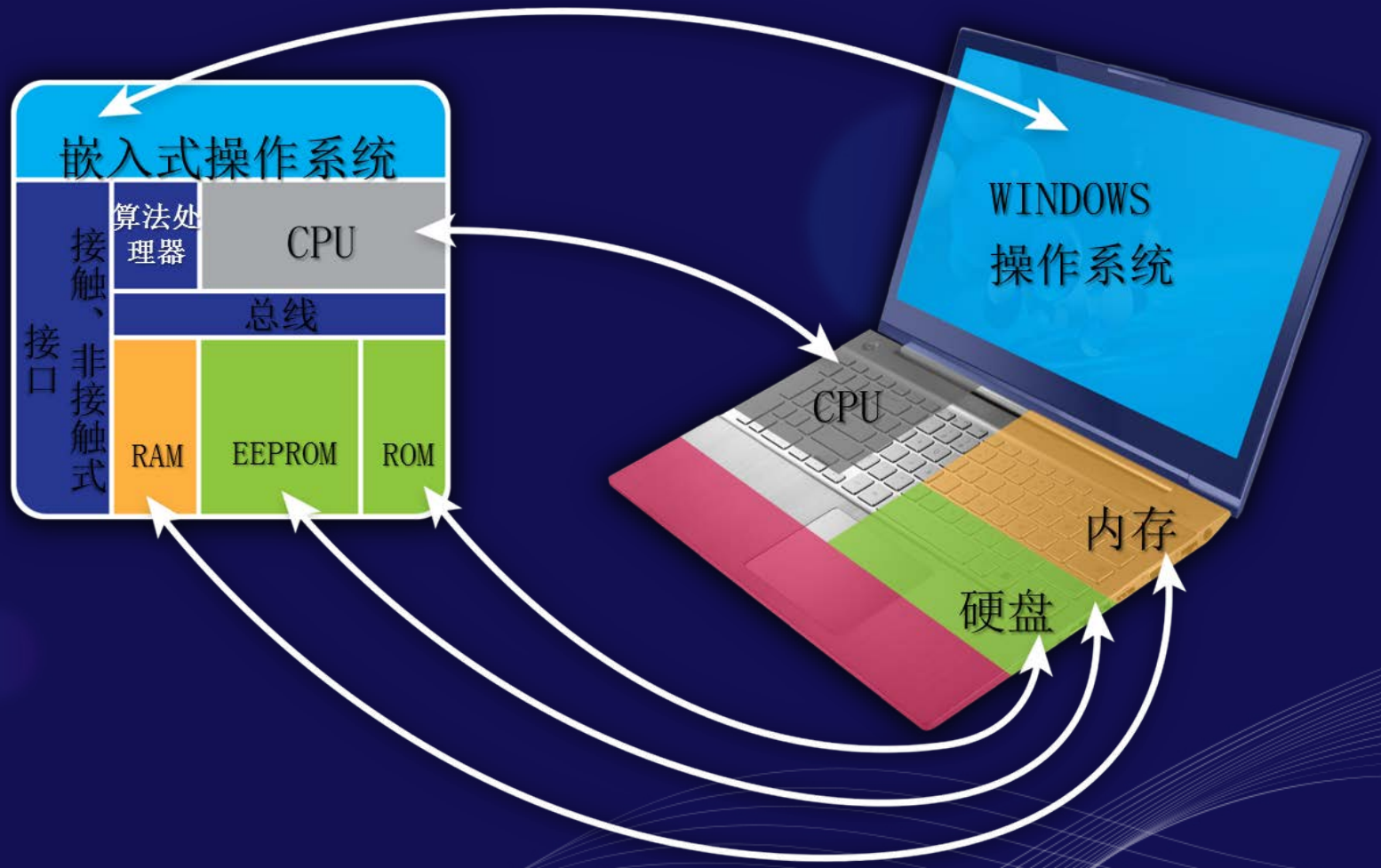
- 基于SE提供的安全存储和安全运算环境，SE可为物联网设备的运营者提供**一个安全的信任根**，由物联网设备运营者发行SE中的设备ID号和证书密钥等，再结合云端的安全云，形成一套完整的物联网安全方案，从而实现可信的身份认证、可靠的通讯加密、数据防篡改和防抵赖，为物联网设备运营者的业务发展保驾护航。

什么是SE芯片

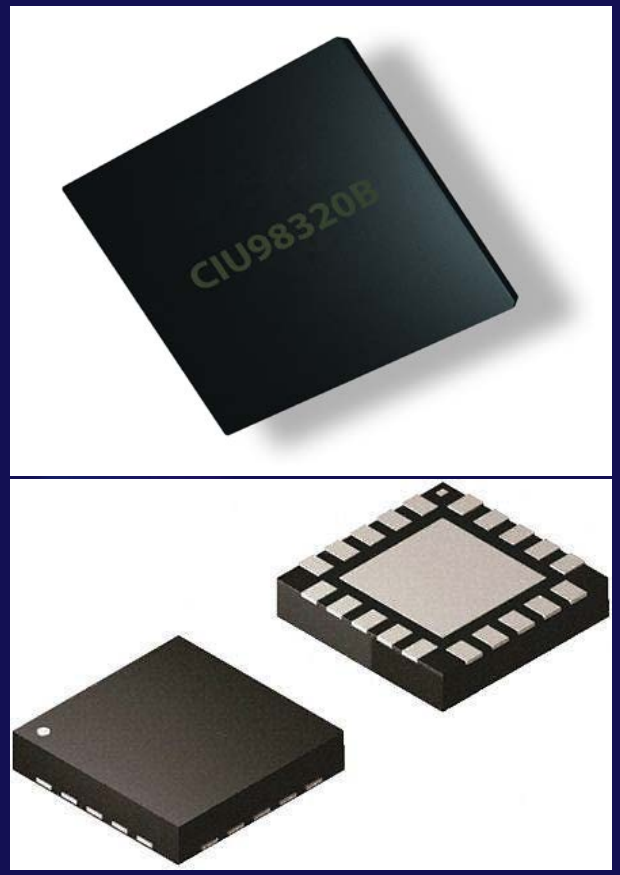
➢SE，Secure Element的简称，直译为安全单元，也称安全芯片。

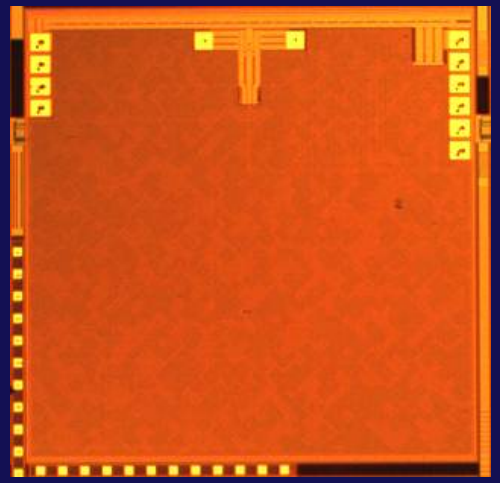
➢安全硬件包括安全的运行环境、安全存储、安全算法、安全接口等；

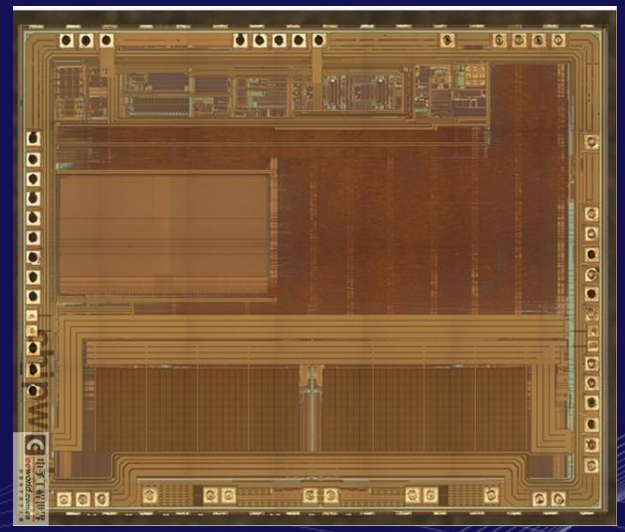
➢安全软件提供安全的交互机制，确保SE与上位机之间命令和数据的交互安全，基于SE对数据进行安全处理、安全计算、安全存储等安全功能



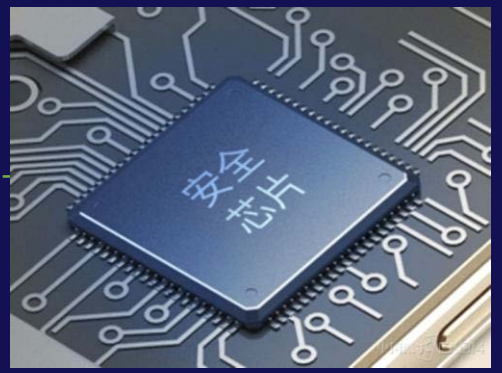
安全芯片-硬件安全







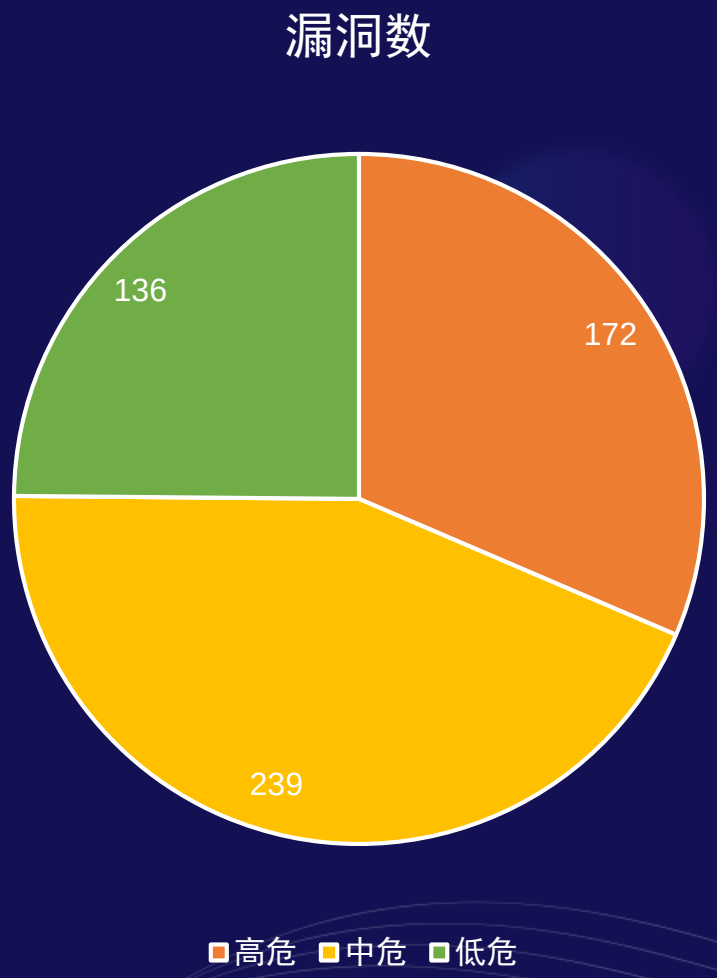
物联网设备安全需求



- **安全存储**
- **加密算法**
- **真随机数**
- 唯一的设备ID号
- 安全密钥存储
- 安全算法：SM1/SM2/SM3/SM4/SM9，DES、RSA、AES、SHA-n、ECC
- 可信的身份认证
- 可靠的通信加密
- 防篡改防抵赖

智能门锁APP平均5.2个高危漏洞

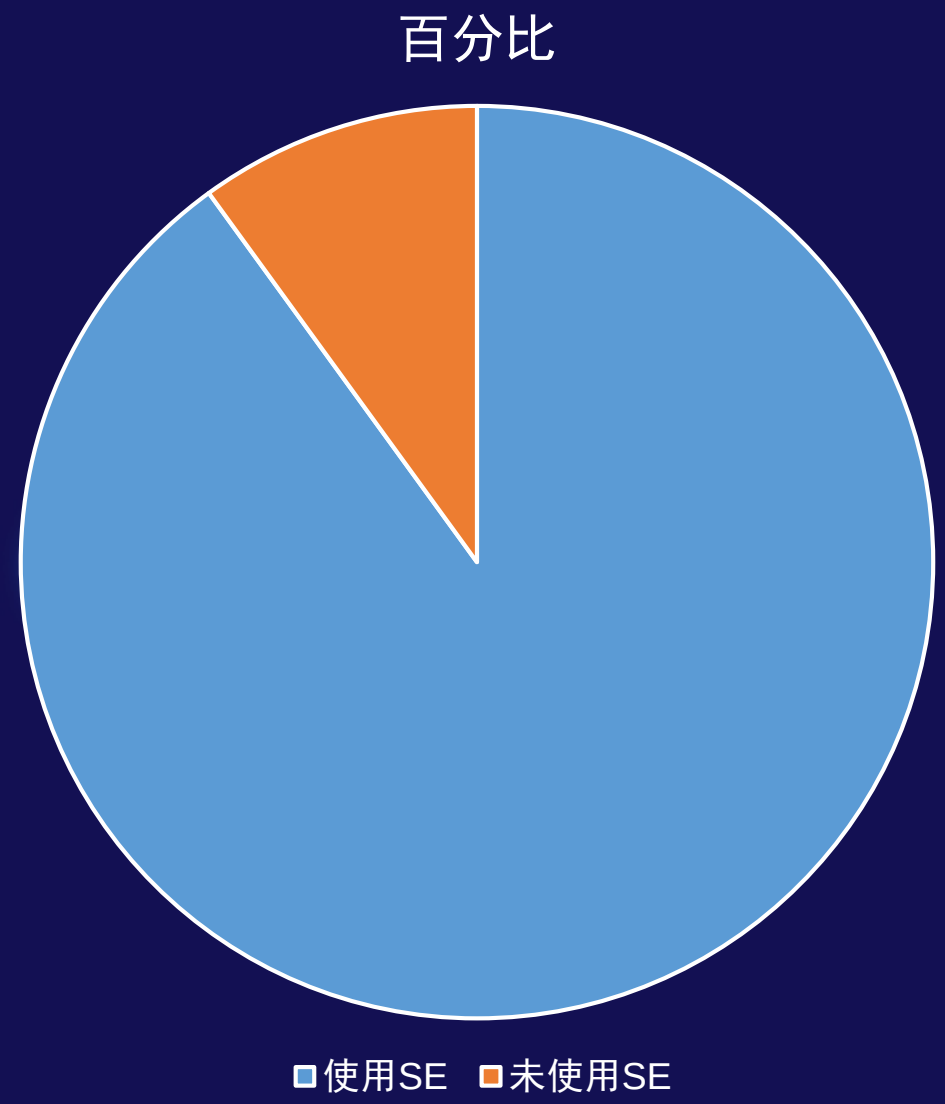
- 测试了33款智能门锁APP
- 共检测出547个漏洞
- 平均每个APP高危漏洞5.2个，中危7.2个，低危4.1个



7大智能门锁设备存在严重通信安全问题

- 明文传输
- 明文密钥存储
- 固定密钥

智能云锁使用SE情况



- 95%以上的智能云锁没有使用SE芯片
- 即便使用了SE，也没有正确利用SE的功能

直接风险

- 中间人劫持，协议破解
- 远程开锁，信息泄露
- 固件升级包篡改

SE芯片具有的特性及应用

常见的安全芯片

常见的安全芯片

1	随机数	真随机数协处理器
2	MD5	MD5摘要算法
3	SHA-1	SHA-1 摘要算法
4	SHA-2	SHA-256，SHA-384，SHA-512 摘要算法
5	SHA-3	SHA3 摘要算法
6	SM3	国密摘要算法 SM3
7	3DES	3DES-112，3DES-168 对称算法
8	AES	AES-128，AES-256 对称算法
9	RSA	RSA1024~RSA2048 非对称算法
10	ECC	椭圆曲线算法，通常256位
11	SM2	国密椭圆曲线算法，算法长度256位
12	SM4	国密对称加密算法
13	SM7	国密对称加密算法
14	SM9	国密非对称数字标识算法

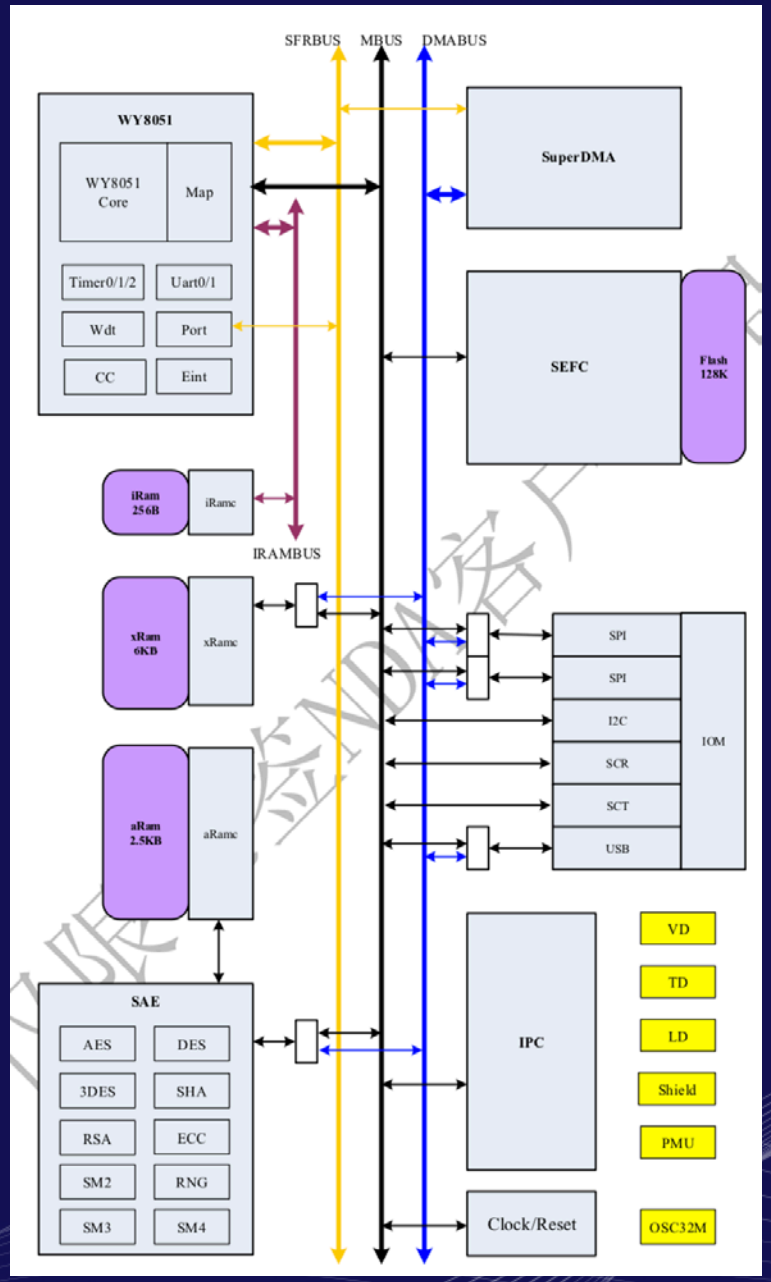
常见的硬件加密引擎

应用多元化

保证任意一种设备以及各种用例（例如支付、票券兑换、交通、访问控制、票务、公司、云计算、电子政务等）中应用程序的安全。根据不同的设备，eSE的功能性趋于多样化，尤其表现在以远程安全的方式检索数据、安全连接、强用户认证、设备整合等方面。

Se芯片系统结构图

- Flash存储内容加密，存储地址加干扰
- SRAM存储内容加密，存储地址加干扰
- 支持存储访问保护（Map）功能，对用户进行分级开发
- 独立用户权限配置管理，不同用户之间仅能调用
- 不能访问彼此数据和程序代码



SE芯片的应用

- 安全是系统工程，没有银弹
- 硬件之外的安全风险点
- SE芯片引入的七寸

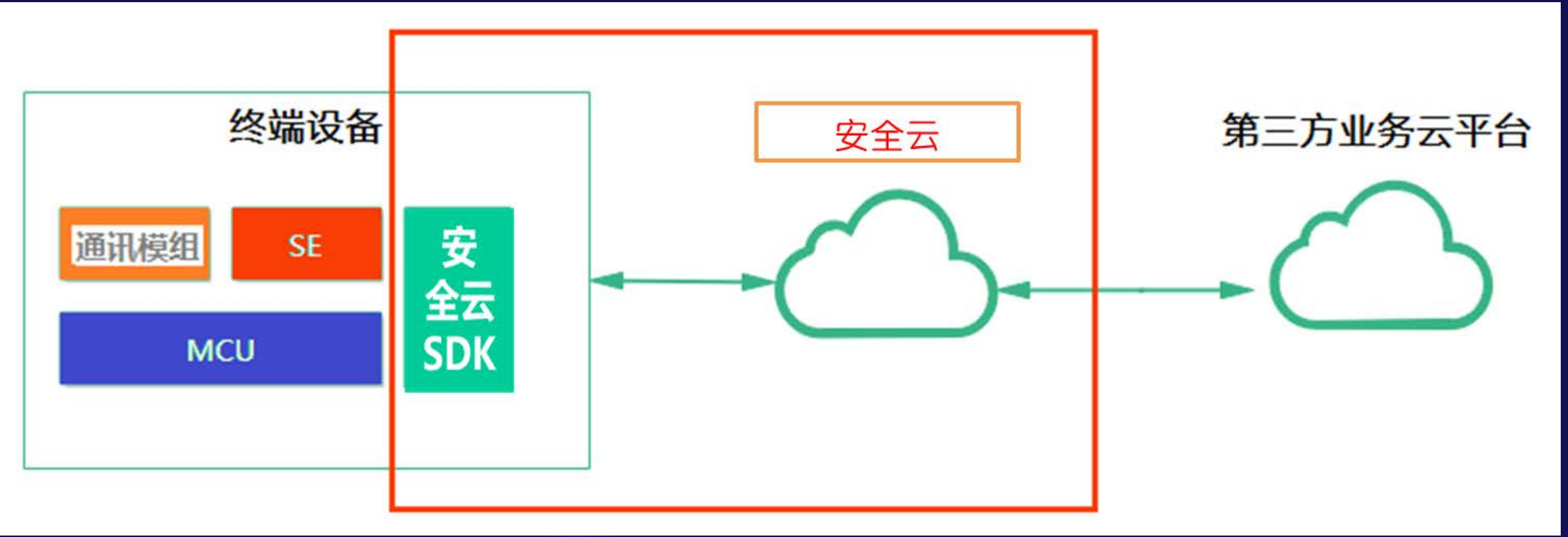
表计行业发展

机械表计

智能表计

物联网表计

物联网SE的应用



唯一设备ID

设备证书

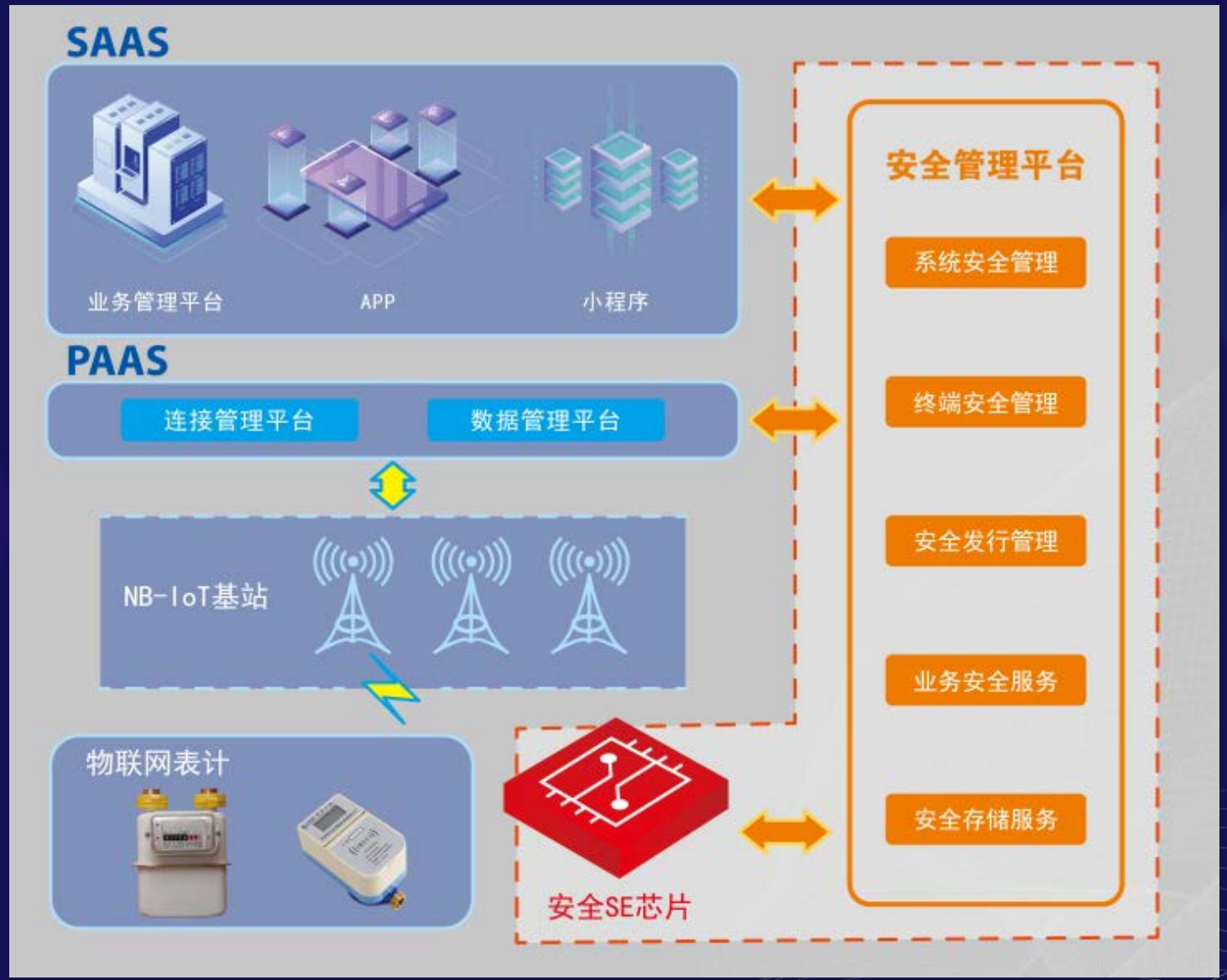
设备私钥

设备ID库

根证书

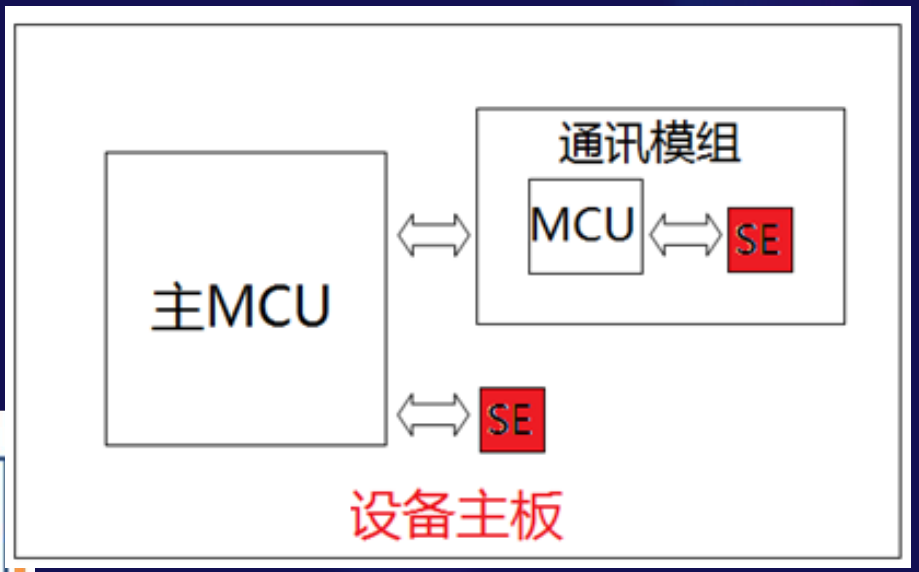
云端私钥

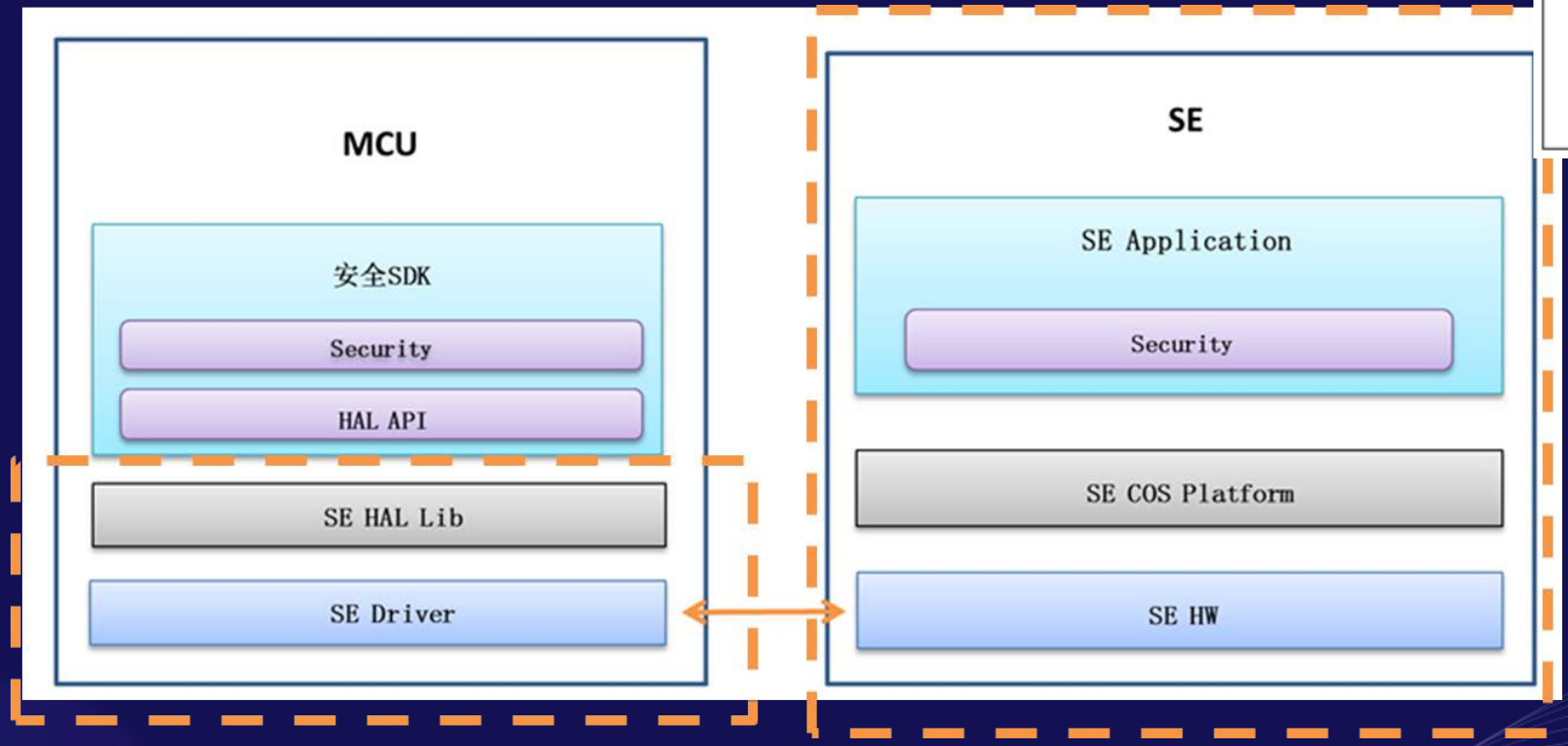
物联网表计安全方案



安全存储设备运营方的密钥等敏感信息，确保设备运营方的数据安全

物联网安全芯片

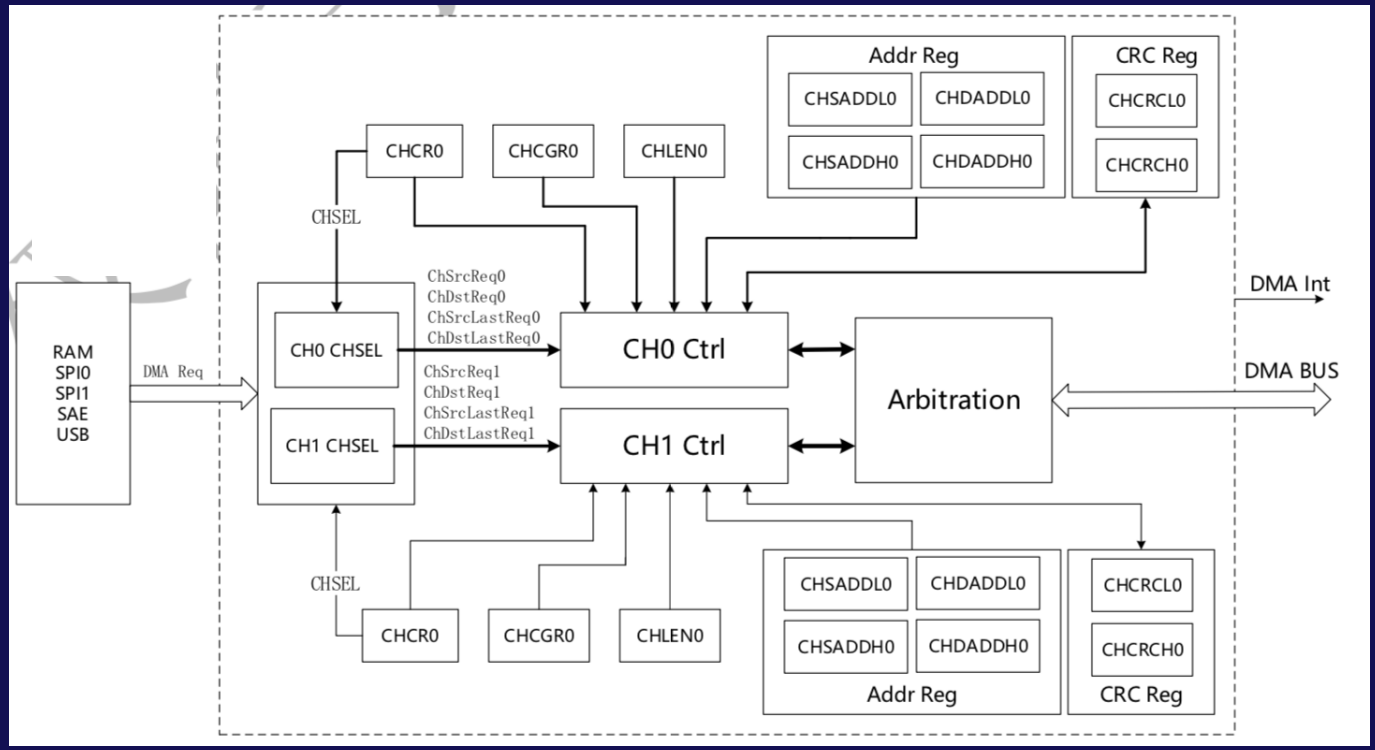

通讯模组
主MCU
MCU
SE
SE
设备主板


MCU
安全SDK
Security
HAL API
SE HAL Lib
SE Driver
SE
SE Application
Security
SE COS Platform
SE HW

SE芯片风险

SDMA

SDMA模块用于在源地址和目的地址之间独立于处理器之间数据搬移。有独立处理器的专用总线DMABUS，数据总线宽为8bit，地址总线位宽为16bit，每次搬移数据长度最大为1024Bytes。可以直接访问系统SRAM空间XRAM，算法SRAM空间ARAM，以及USB，SPI及SAE模块的FIFO空间

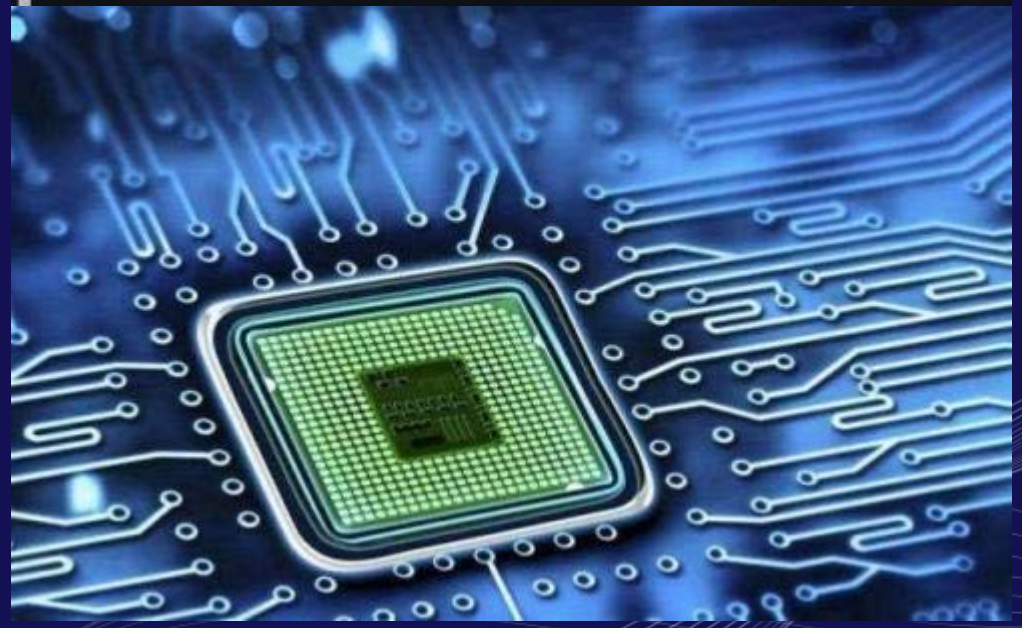


SDMA内部结构

SDMA溢出攻击

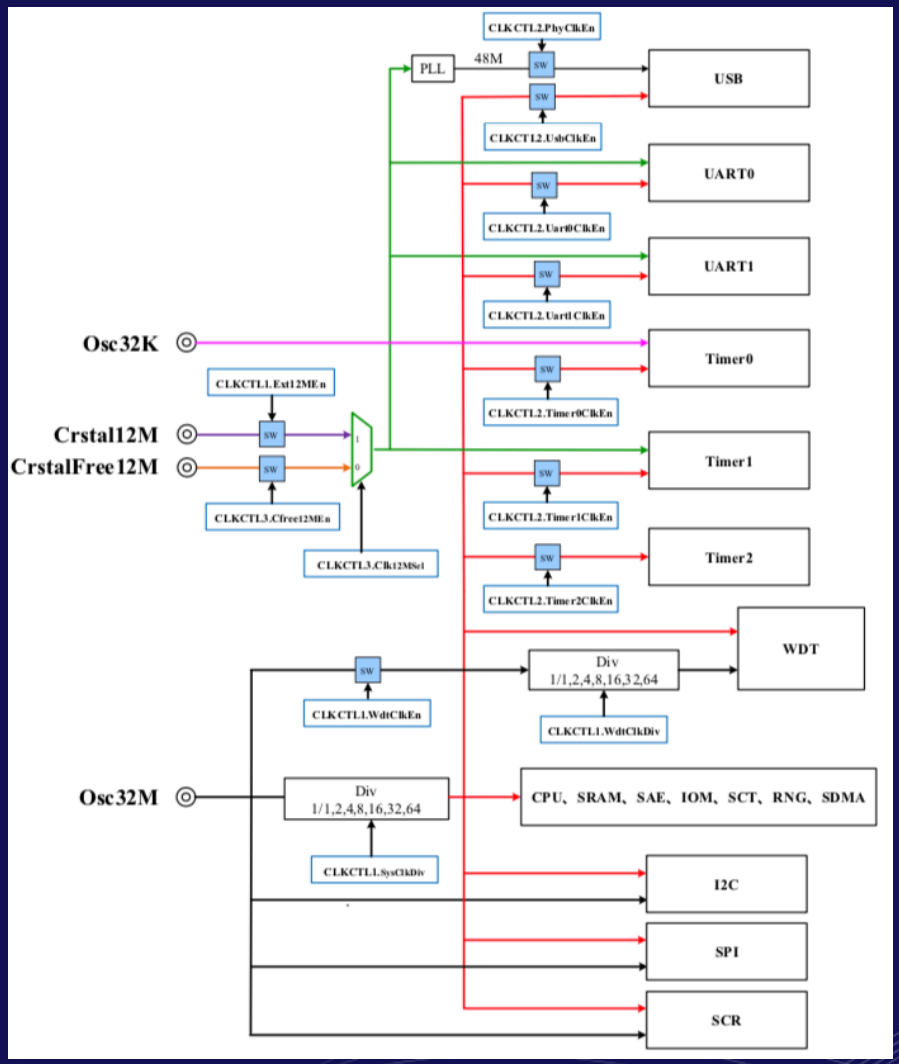
通过对物理通道CH0和CH1，通道的配置寄存器控制需改传输数据长度寄存器CHLEN的长度限制，进行溢出漏洞的触发从而获得SDMA攻击的权限

```
meltdown@meltdown: ./meltdown
e01d8110: 61 78 20 6f 72 20 73 74  61 74 65 20 6d 61 63 68 |ax or state mach|
e01d8120: 69 6e 65 2c 20 69 74 20  69 73 20 62 65 69 6e 67 |ine, it is being|
e01d8130: 20 75 73 65 64 20 77 69  74 68 20 61 75 74 68 6f | used with autho|
e01d8140: 72 69 7a 61 74 69 6f 6e  20 66 72 6f 6d 0a 20 53 |rization from. S|
e01d8150: 69 6c 69 63 6f 6e 20 47  72 61 70 68 69 63 73 2c |ilicon Graphics,|
e01d8160: 20 49 6e 63 2e 20 20 48  6f 77 65 76 65 72 2c 20 | Inc.  However, |
e01d8170: 74 68 65 20 61 75 74 68  6f 72 73 20 6d 61 6b 65 |the authors make|
e01d8180: 20 6e 6f 20 63 6c 61 69  6d 20 74 68 61 74 20 4d | no claim that M|
e01d8190: 65 73 61 0a 20 69 73 20  69 6e 20 61 6e 79 20 77 |esa. is in any w|
e01d81a0: 61 79 20 61 20 63 6f 6d  70 61 74 69 62 6c 65 20 |ay a compatible |
e01d81b0: 72 65 70 6c 61 63 65 6d  65 6e 74 20 66 6f 72 20 |replacement for |
e01d81c0: 4f 70 65 6e 47 4c 20 6f  72 20 61 73 73 6f 63 69 |OpenGL or associ|
e01d81d0: 61 74 65 64 20 77 69 74  68 0a 20 53 69 6c 69 63 |ated with. Silic|
e01d81e0: 6f 6e 20 47 72 61 70 68  69 63 73 2c 20 49 6e 63 |on Graphics, Inc|
e01d81f0: 2e 0a 20 2e 0a 20 54 68  69 73 20 76 65 72 73 69 |.. .. This versi|
e01d8200: 6f 6e 20 6f 66 20 4d 65  73 61 20 70 72 6f 76 69 |on of Mesa provi|
e01d8210: 64 65 73 20 47 4c 58 20  61 6e 64 20 44 52 49 20 |des GLX and DRI |
e01d8220: 63 61 70 61 62 69 6c 69  74 69 65 73 3a 20 69 74 |capabilities: it|
e01d8230: 20 69 73 20 63 61 70 61  62 6c 65 20 6f 66 0a 20 | is capable of. |
e01d8240: 62 6f 74 68 20 64 69 72  65 63 74 20 61 6e 64 20 |both direct and |
e01d8250: 69 6e 64 69 72 65 63 74  20 72 65 6e 64 65 72 69 |indirect renderi|
```



时钟

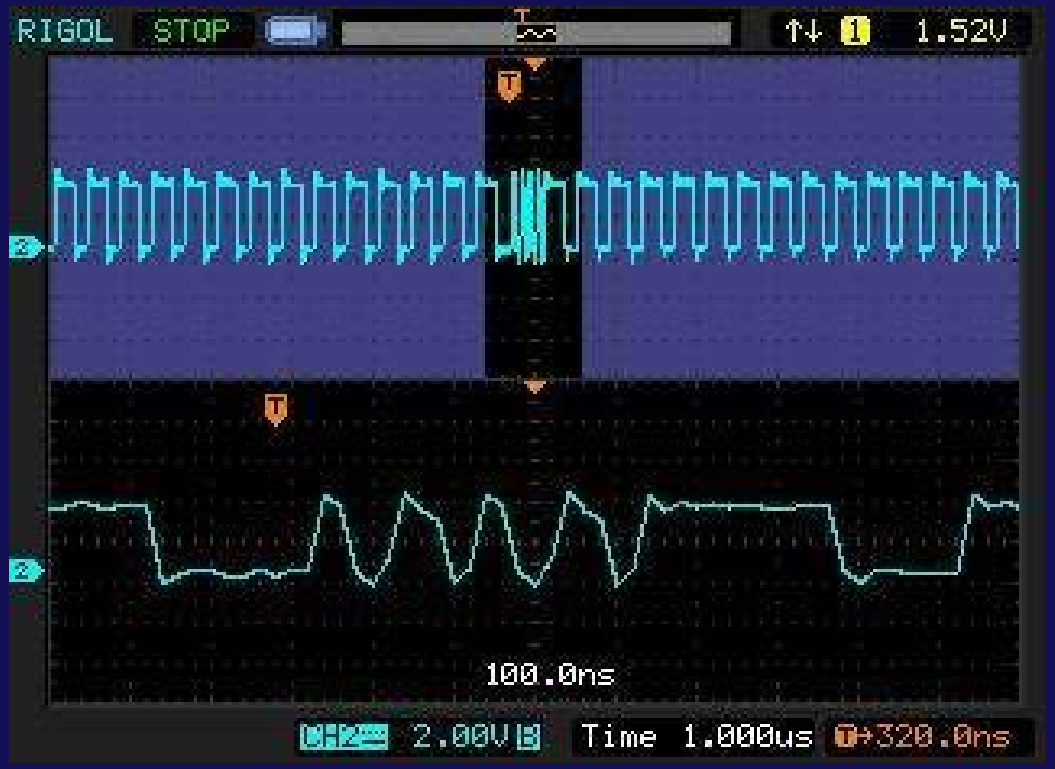
SDMA模块用于在源地址和目的地址之间独立于处理器之间数据搬移。有独立处理器的专用总线DMABUS，数据总线宽为8bit，地址总线位宽为16bit，每次搬移数据长度最大为1024Bytes。可以直接访问系统SRAM空间XRAM，算法SRAM空间ARAM，以及USB，SPI及SAE模块的FIFO空间



时钟内部结构

时钟攻击

错误注入攻击指的是在计算设备中，故意引入毛刺，以期望改变软件硬件逻辑的执行流。错误注入一般期望有两种效果，避免执行和破坏正在处理的数据，这些可以用来绕过安全认证和泄露密码学算法的密钥



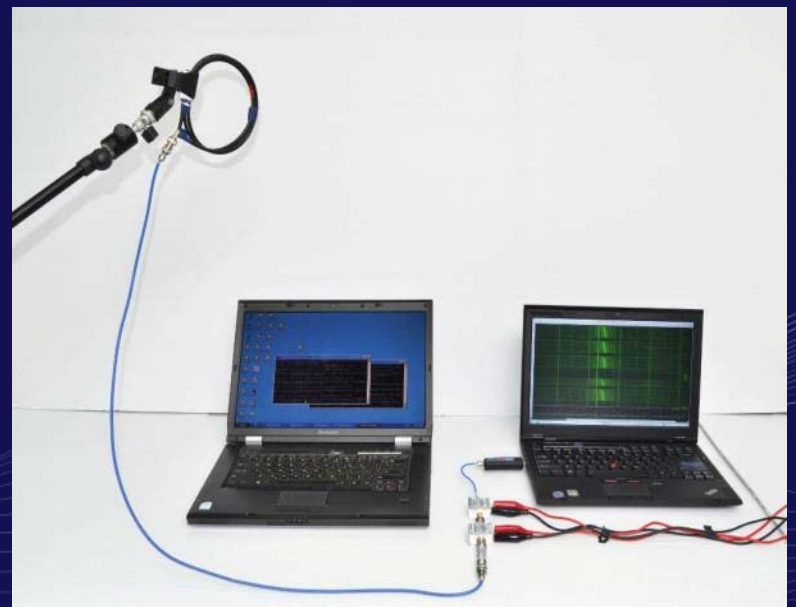
时钟攻击

侧信道攻击

传统密码分析学认为一个密码算法在数学上安全就绝对安全，这一思想被Kelsey等学者在1998年提出的侧信道攻击(Side-channel Attacks,SCA)理论所打破。

侧信道攻击与传统密码分析不同,侧信道攻击利用功耗、电磁辐射等方式所泄露的能量信息与内部运算操作数之间的相关性，通过对所泄露的信息与已知输入或输出数据之间的关系作理论分析,选择合适的攻击方案,获得与安全算法有关的关键信息。

目前侧信道理论发展越发迅速，从最初的简单功耗分析（SPA）到多阶功耗分析（CPA），碰撞攻击、模板攻击、电磁功耗分析以及基于人工智能和机器学习的侧信道分析方式，侧信道攻击方式也推陈出新，从传统的直接能量采集发展到非接触式采集、远距离采集、行为侧信道等等。

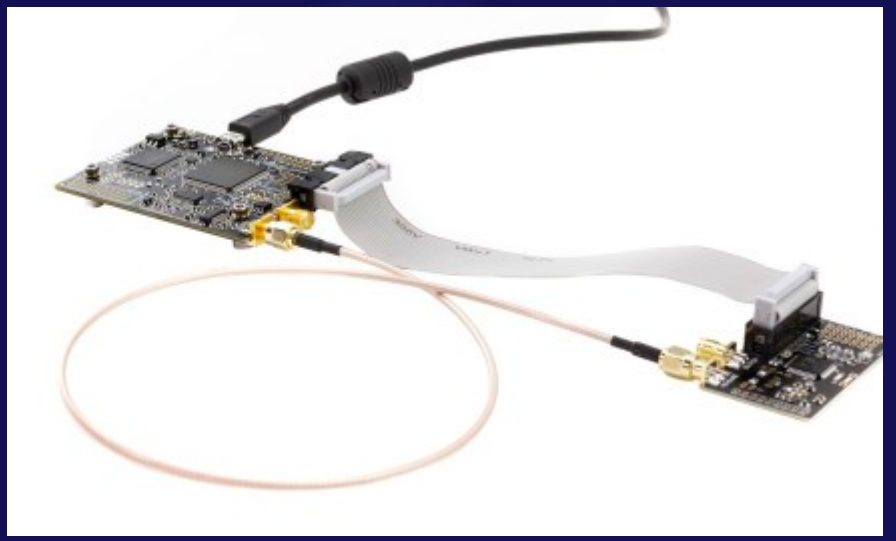


侧信道攻击

Kelsey等学者在1998年提出了侧信道攻击(Side-channel Attacks)，这种攻击方法与传统密码分析不同。它是针对加密硬件在运行过程中的时间消耗、功耗、电磁辐射等侧信道信息泄露而对加密设备进行攻击的方法。

侧信道——攻击工具

- 采用成本仅为2000多人民币的攻击工具CW1173 ChipWhisperer-Lite Part2来展示侧信道攻击方法。硬件包括：多功能功率分析采集板和目标板。



- 信号采集板基于FPGA实现，带有10位ADC，采样率为105ms/s，加上+55db的增益放大器，可以测量小信号。
- 目标板为某门锁使用的未通过EAL4+认证的SE+MCU。

侧信道能量攻击原理

芯片物理结构是CMOS电路组合而成，CMOS 电路根据输入的不同电信号动态改变输出状态，实现0或1的表示，完成相应的运算。

运算器的静态功率取决于芯片内的晶体管数量和布局。动态功率取决于芯片内流动的数据。每次一个比特从0变为1（反之亦然），都需要一些电流来给数据线充（放）电。这种动态功率的变化使得我们可以了解运算器内部发生了什么。

侧信道能量攻击方法-相关性能量分析（CPA）

CPA攻击技术利用了统计学中的皮尔逊相关系数进行分析攻击，其攻击过程：

1）运算AES算法的芯片中，将不同的明文数据在AES密钥作用下进行加密运算，通过监听获取芯片的能耗，记为T 。

2）通过猜测密钥，产生相应的中间值，根据中间值的汉明重量或汉明距离，计算得到假设能量消耗，记为 D。

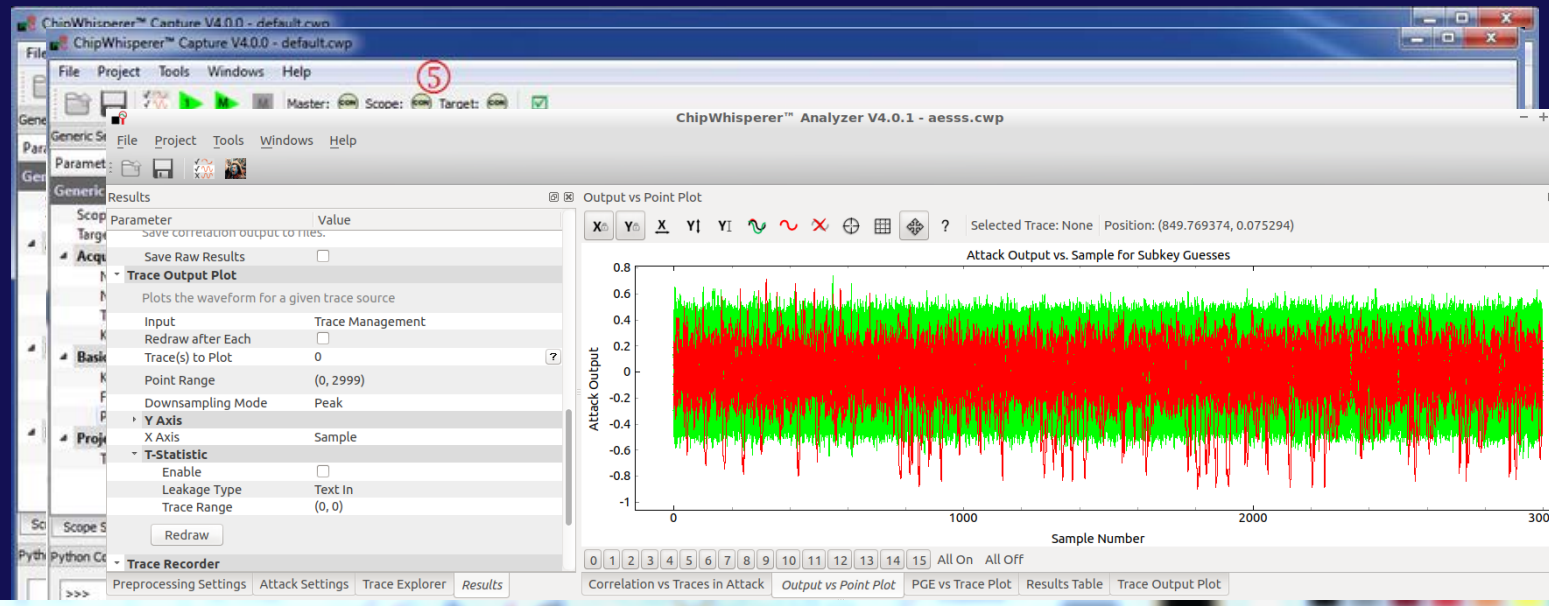
3）根据下式计算假设能量消耗与实测能量迹的线性相关系数。

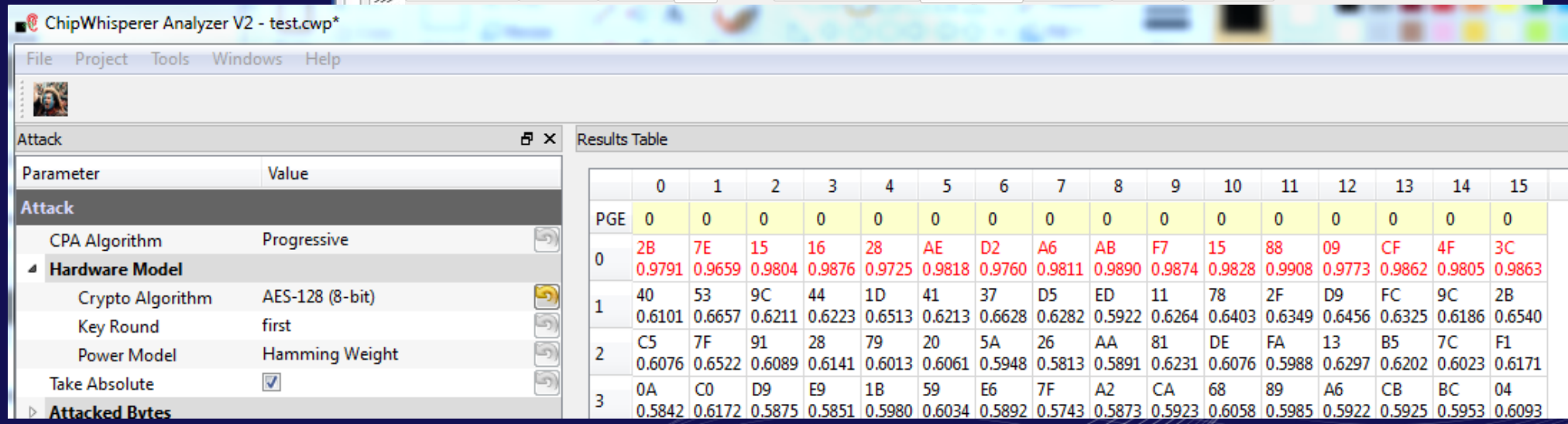
$$\rho(T,D)=(E(T\bullet D)-E(T)E(D))/\sqrt{(Var(T)Var(D))}$$

式（2）中 E()表示求平均值， Var()表示求方差。 ρ的范围在[-1,1]之间，当 ρ取绝对值的最大值时，即假设能量消耗与真实测量的能量迹线性相关性（即相关性系数 ρ ）达到最大，则此时D所对应的猜测的密钥即为正确的密钥。

侧信道能量攻击方法 –软件

工具软件提供能量波形采集端口，通过连接 板上的 SMA 接口，就可以对能量波形进行采集，利用开源软件进行分析，可以实现简单能量分析、CPA攻击、模板攻击等。





	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
PGE	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	2B 0.9791	7E 0.9659	15 0.9804	16 0.9876	28 0.9725	AE 0.9818	D2 0.9760	A6 0.9811	AB 0.9890	F7 0.9874	15 0.9828	88 0.9908	09 0.9773	CF 0.9862	4F 0.9805	3C 0.9863
1	40 0.6101	53 0.6657	9C 0.6211	44 0.6223	1D 0.6513	41 0.6213	37 0.6628	D5 0.6282	ED 0.5922	11 0.6264	78 0.6403	2F 0.6349	D9 0.6456	FC 0.6325	9C 0.6186	2B 0.6540
2	C5 0.6076	7F 0.6522	91 0.6089	28 0.6141	79 0.6013	20 0.6061	5A 0.5948	26 0.5813	AA 0.5891	81 0.6231	DE 0.6076	FA 0.5988	13 0.6297	B5 0.6202	7C 0.6023	F1 0.6171
3	0A 0.5842	C0 0.6172	D9 0.5875	E9 0.5851	1B 0.5980	59 0.6034	E6 0.5892	7F 0.5743	A2 0.5873	CA 0.5923	68 0.6058	89 0.5985	A6 0.5922	CB 0.5925	BC 0.5953	04 0.6093

AES加密逻辑

```
state = in
AddRoundKey(state, w[0, Nb-1])
for round = 1 step 1 to Nr - 1
    SubBytes(state)                    // <-- 有效的攻击位置，选择第一轮!
    ShiftRows(state)
    MixColumns(state)
    AddRoundKey(state, w[round*Nb, (round+1)*Nb-1])
end for
SubBytes(state)
ShiftRows(state)
AddRoundKey(state, w[Nr*Nb, (Nr+1)*Nb-1])
out = state
```

不合规的开发习惯

开发人员滥用加密引擎接口

开发人员未开启相关芯片保护机制

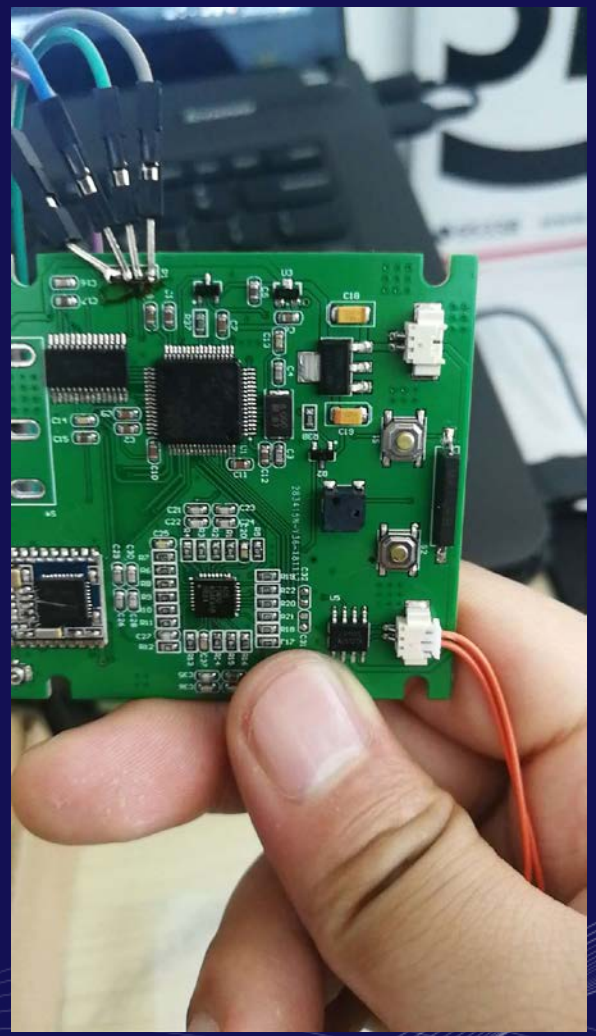
电路设计人员未对Se进行防护保护设计

开发人员寄存器调用不合理

二：某智能门锁项目中的Se使用

ATECC508A加密芯片主要功能为：

- 经过优化的密钥存储和认证功能
- 应用所存储的私钥进行ECDH操作
- ECDSA（椭圆曲线数字签名算法）签名与验证
- 支持X.509认证格式
- 256位SHA/HMAC硬件引擎

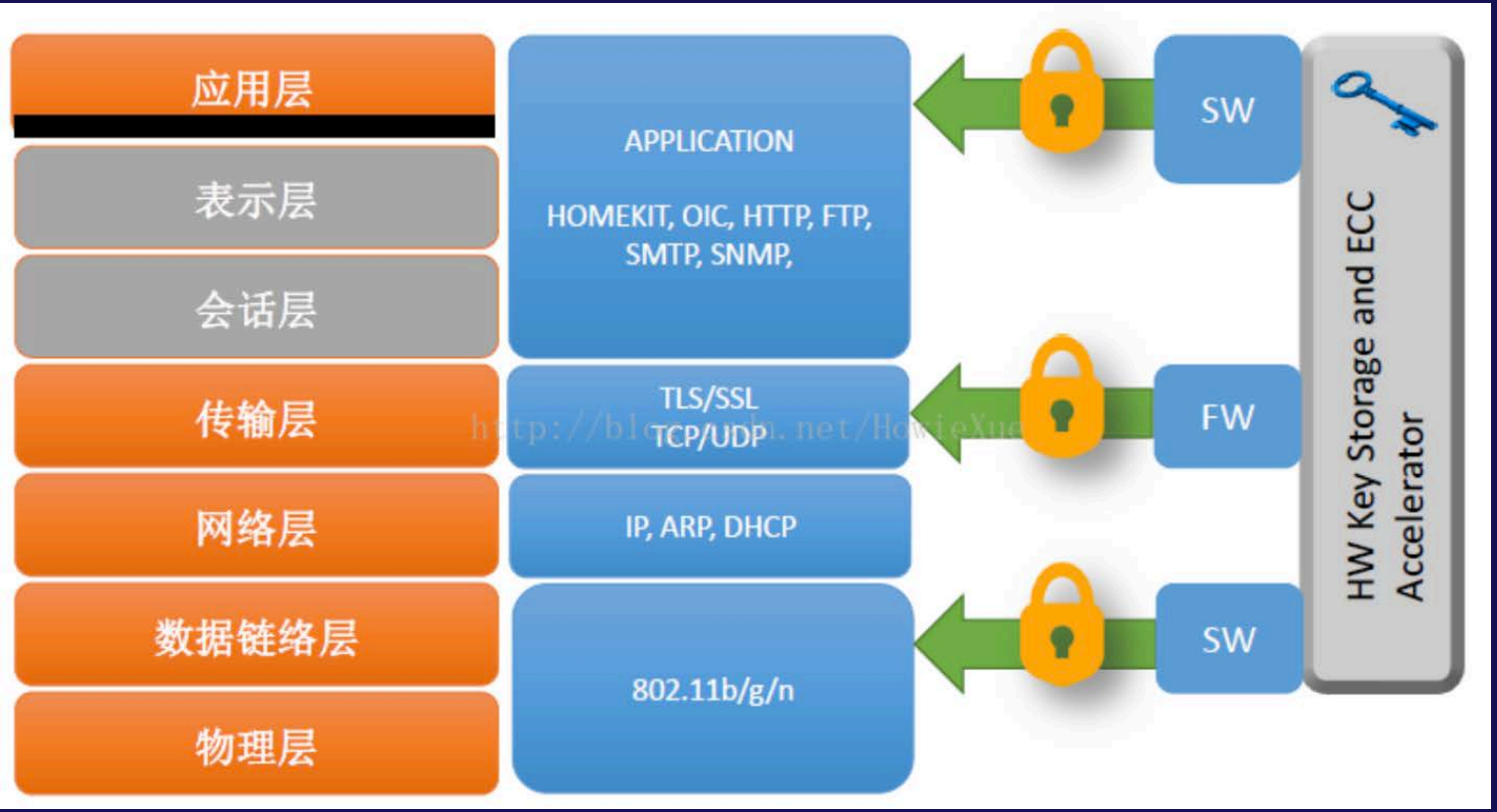


Se芯片通过IIC接口进行传输信息

根据芯片手册说明芯片对传输数据不做加密操作进行明文传输。

通过IIC读取设备进行Se芯片的通路信息抓取并进行内存的调用操作。

改进意见



在进行芯片选型及开发的过程中一定要对数据手册熟读，进行合理的芯片选型

建议使用ATECC608（成本无变化，整体安全性防护提升）

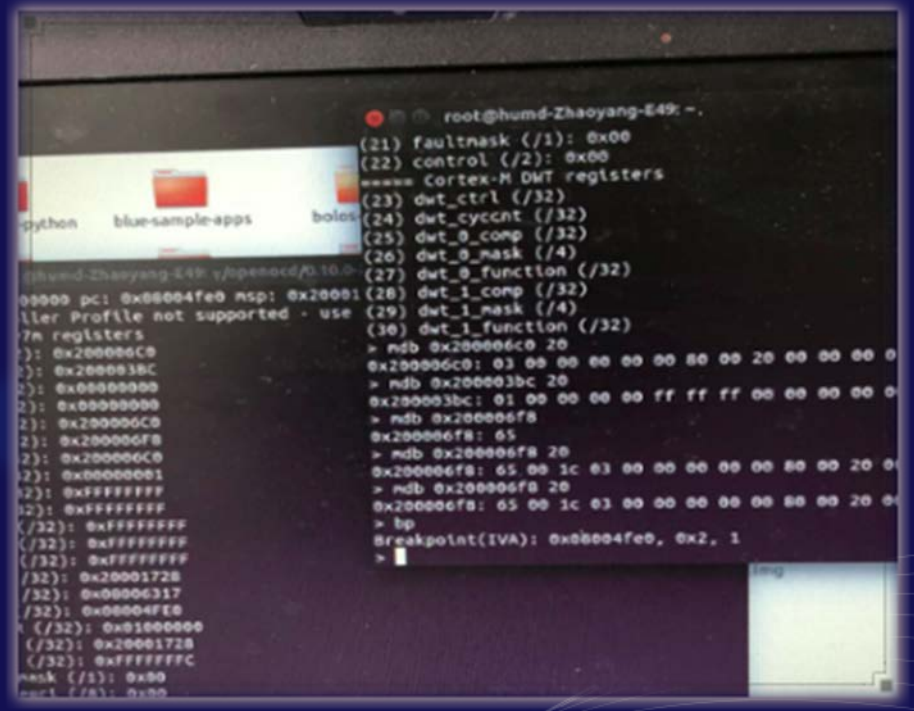
某安全产品中的SE使用

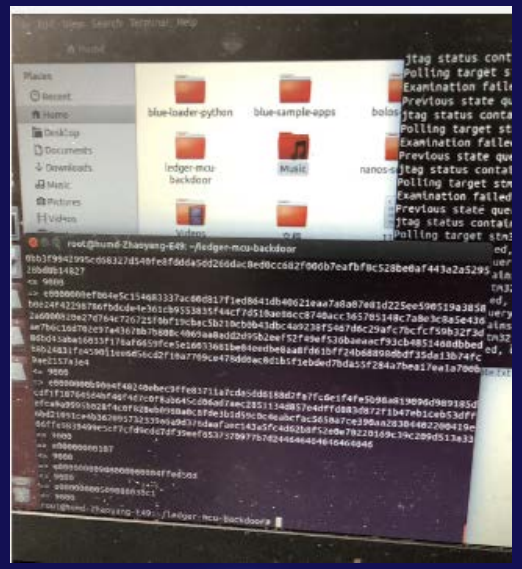
ST官方系列加密芯片主要功能为：

- 提供加密引擎及通信加密中的相关功能
- 提供关键信息的存储

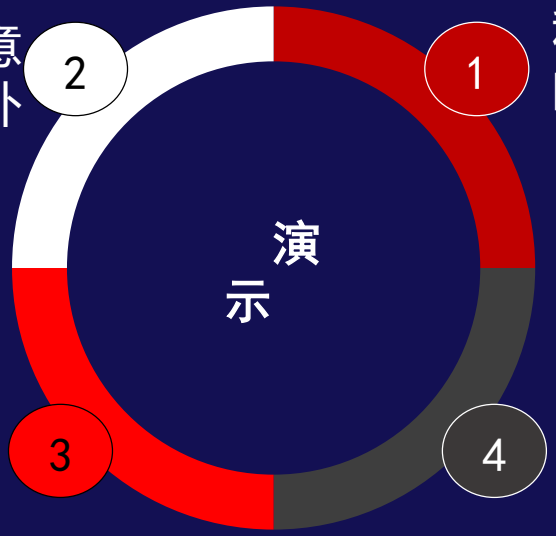
主要的测试手段通过电路预留测试

接口JTAG进行电路的控制及调试

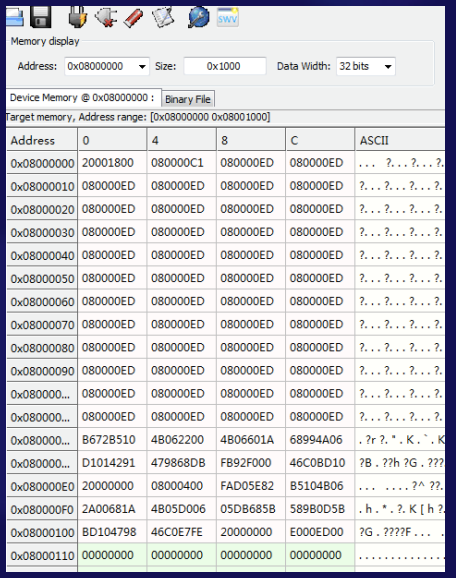




通过升级写入恶意固件，不用破坏外壳



利用jtag读出完整的固件



Address	0	4	8	C	ASCII
0x08000000	20001800	080000C1	080000ED	080000ED	... ?...?...?.
0x08000010	080000ED	080000ED	080000ED	080000ED	?...?...?...?.
0x08000020	080000ED	080000ED	080000ED	080000ED	?...?...?...?.
0x08000030	080000ED	080000ED	080000ED	080000ED	?...?...?...?.
0x08000040	080000ED	080000ED	080000ED	080000ED	?...?...?...?.
0x08000050	080000ED	080000ED	080000ED	080000ED	?...?...?...?.
0x08000060	080000ED	080000ED	080000ED	080000ED	?...?...?...?.
0x08000070	080000ED	080000ED	080000ED	080000ED	?...?...?...?.
0x08000080	080000ED	080000ED	080000ED	080000ED	?...?...?...?.
0x08000090	080000ED	080000ED	080000ED	080000ED	?...?...?...?.
0x080000...	080000ED	080000ED	080000ED	080000ED	?...?...?...?.
0x080000...	080000ED	080000ED	080000ED	080000ED	?...?...?...?.
0x080000...	B672B510	4B062200	4B06601A	68994A06	.?r?.*.K.`.K
0x080000...	D1014291	479868DB	FB92F000	46C0BD10	?B.??h?G.???
0x080000E0	20000000	08000400	FAD05E82	B5104B06	... .?^ ??.
0x080000F0	2A00681A	4B05D006	05DB685B	589B0D5B	.h.*.?.K[h?.
0x08000100	BD104798	46C0E7FE	20000000	E000ED00	?G.????F... .
0x08000110	00000000	00000000	00000000	00000000	.....

写入后显示

记录pin码， 现场演示

改进意见

- 关闭物理调试接口
- 对内部芯片调试接口进行调用防护
- 开发增加防护报警机制

SE存在的风险

- 认证类加密芯，其优点是加密芯片平台安全，算法统一，应用简单。
- 缺点是整体加密方案安全性较低，对板上主控MCU的保护力度较弱，已经证明存在明显安全漏洞。是可以通过对MCU的攻击，间接破解掉加密芯片的。

- TEE进阶
- 将板上主控MCU的程序和数据移植一部分到加密芯片中运行，借助加密芯片完成MCU缺失的功能，同时又保证这部分程序的绝对安全，进而保证整个产品的安全性。

TEE系统的现状

什么是TEE

它是移动设备主处理器上一个安全区域，与移动OS并行存在，提供一个隔离的执行环境，保证隔离执行、可信应用的完整性、可信数据的机密性、安全存储等。

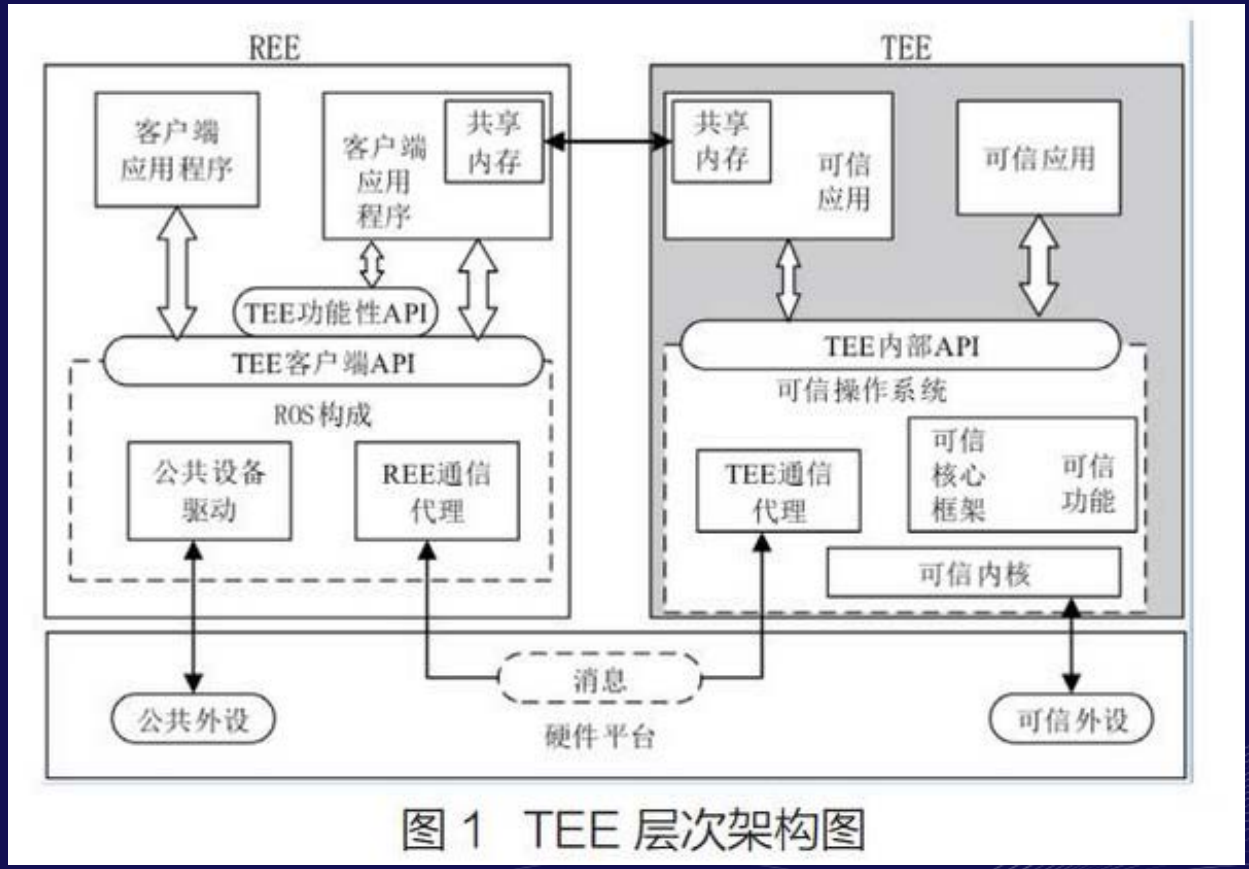


图 1　TEE 层次架构图

TEE的发展

01

2009年

- OMTP(Open Mobile Terminal Platform)，首次定义了TEE：“一组软硬件组件，可以为应用程序提供必要的设施”。

02

2010年

- GSMA(Global System for Mobile Communications Alliance）开始主持OMTP标准及TEE。

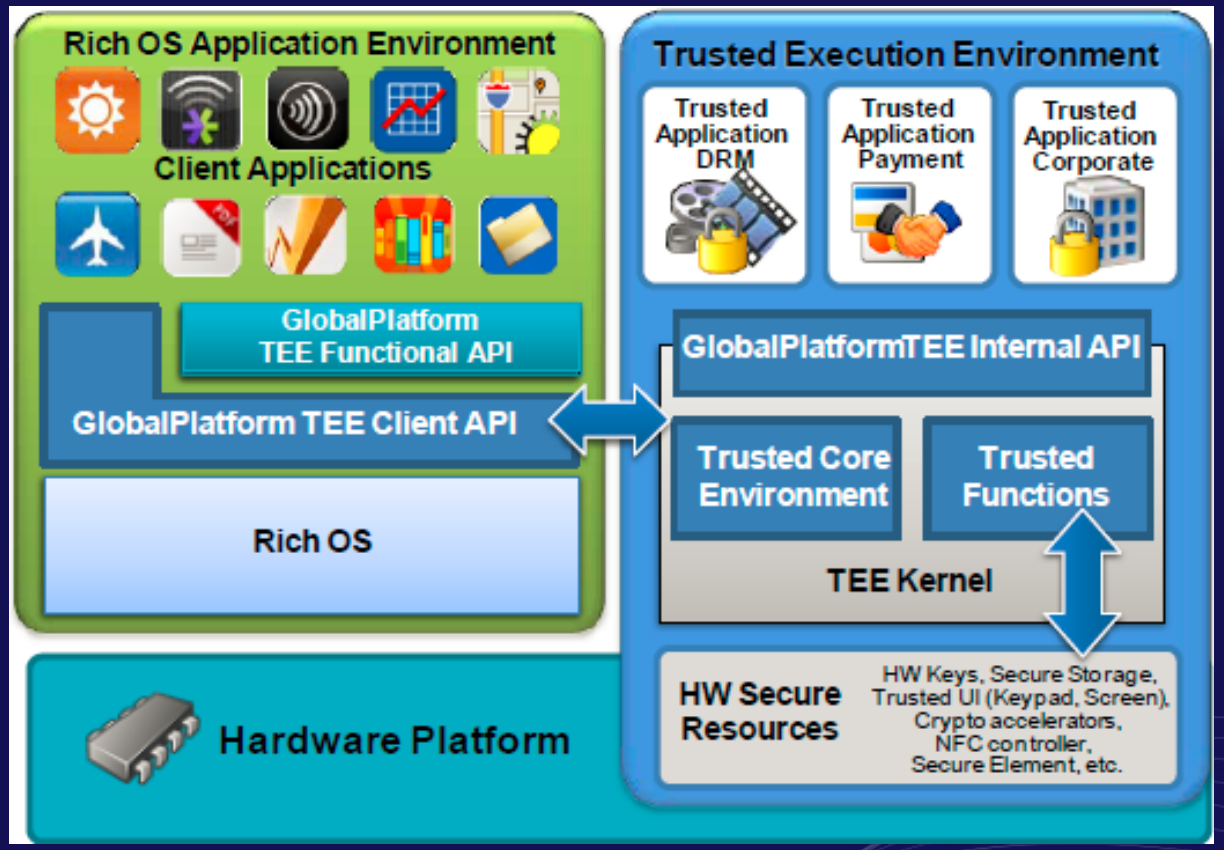
03

2010年

- GSMA宣布自己的TEE标准，构成了目前TEE的基础。包括TEE client API、TEE internal API、一整套TEE系统体系。

TEE的应用现状

目前TEE不仅在手机上应用，而且在越来越多的PC和笔记本也开始使用。且在云平台，TEE也得到了越来越多的重视。



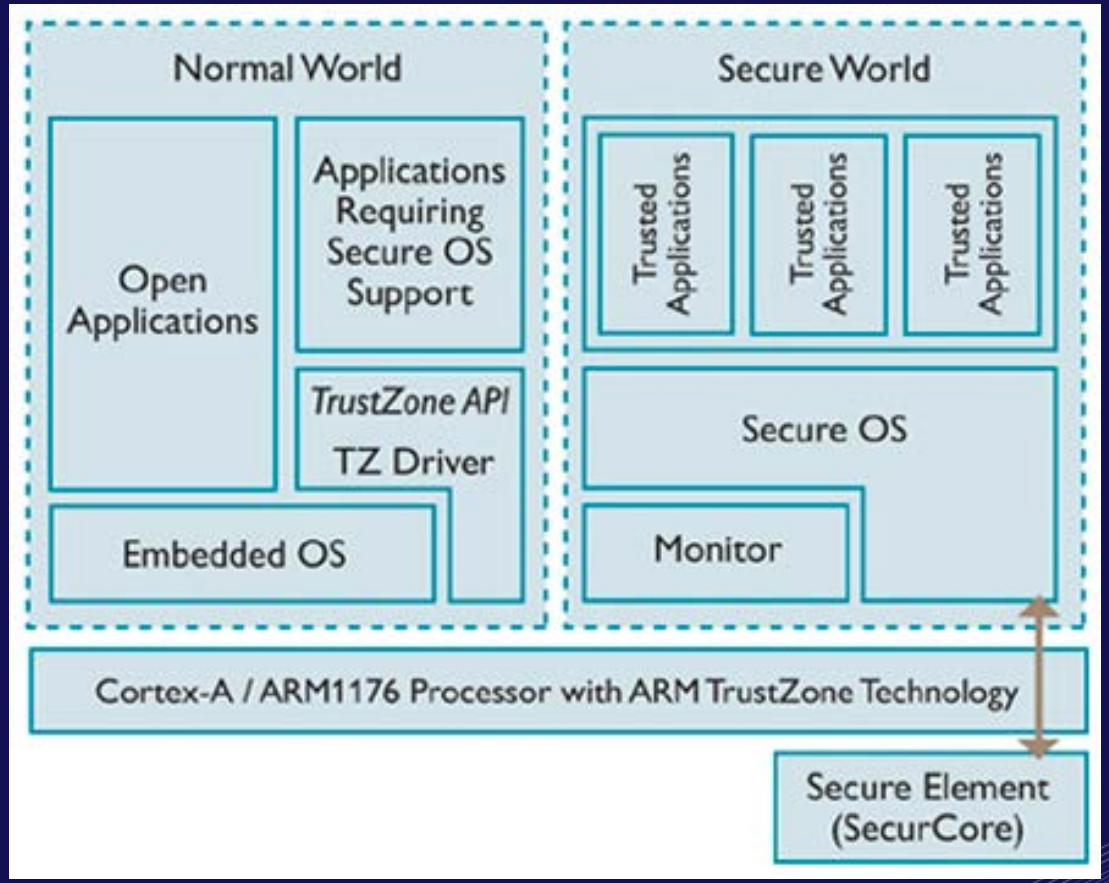
TEE系统的重大问题

设计问题：

1）可信计算基过大。

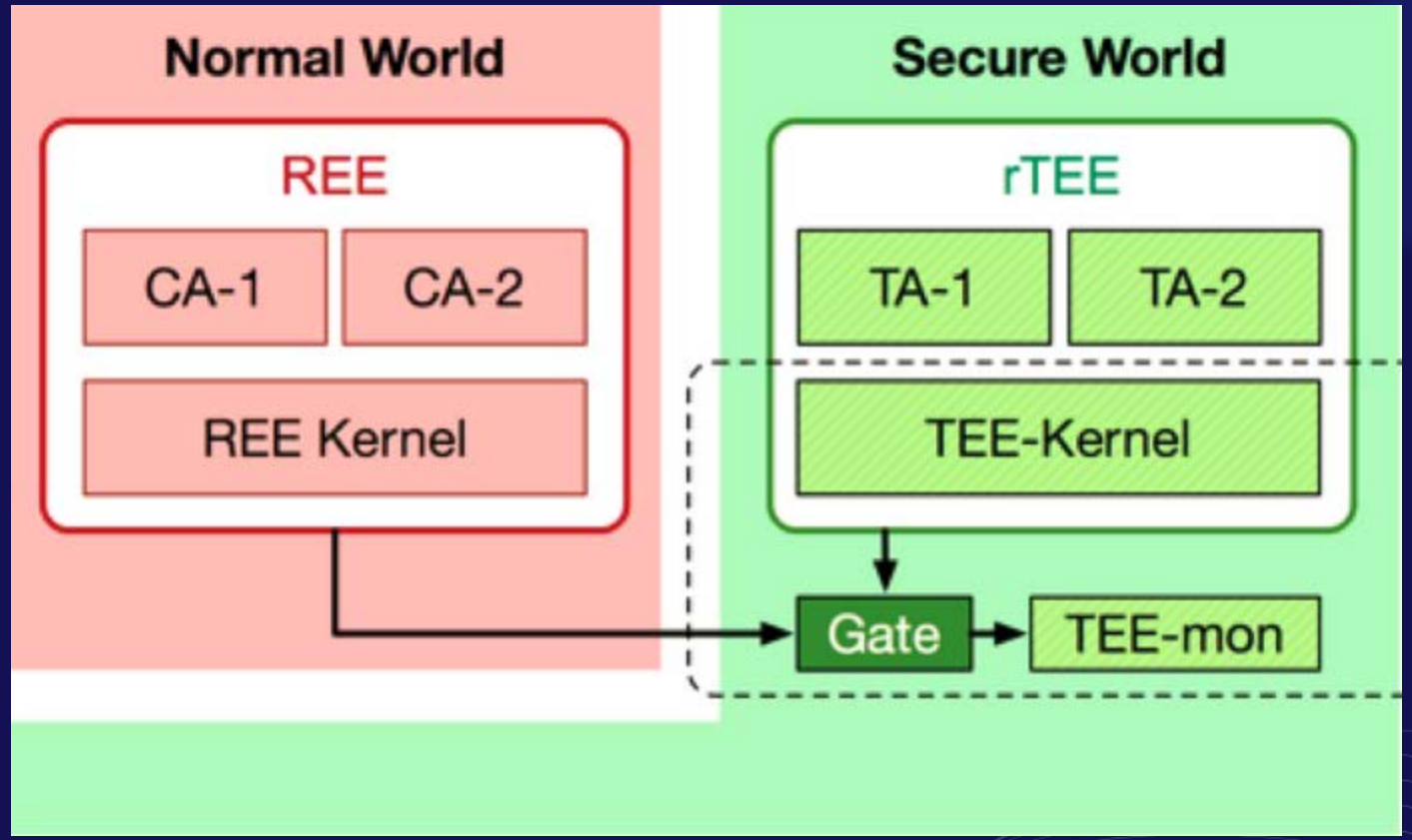
2）可信用户交互支出不足。

3）与普通操作系统的交互的安全隐患。



启动环境

当前绝大部分的TA和CA之间的交互依赖于操作系统，如果操作系统被攻破，那么任何CA都可以和TA建立连接。



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

– 固件刷机攻击

- DMA攻击

载体攻击

载体攻击的目标是通过导出数据载体的数据至另一台设备中，以复制其中的数据。

- 数据暴露原因：

– 数据非安全删除

– OS的数据缓冲区

– 数据缓存机制

– 内存泄露

Bootroom攻击

远程启动服务截取带有RPL的网络接口卡发出引导记录请求的广播（broadcasts），重新创建服务器建立一个连接来响应它，并加载恶意MS-DOS启动文件到工作站的内存中。

固件回滚攻击（rollback）

固件回滚攻击和固件刷机攻击的目标一致，但是实现方式不同，固件刷机攻击是通过物理手段取下存储芯片刷入固件，而固件回滚攻击是通过利用固件防回滚机制的漏洞，刷入低版本存在漏洞的固件。

TEE不能绝对解决的问题

现在ARM TEE主要还是以TrustZone作为支撑。但TrustZone依然有待提高，比如它没有内存加密，可扩展性有待提高， 用多个TrustZone或多个TEE现在还做不到。而且，TEE存在单一故障点问题，TEE出了错整个系统的安全性就没了。

TEE与上层安全应用自身的安全性更需要重视

需要从整体设计与安全评估进行TEE安全增强

现状与趋势	问题与挑战
1、芯片厂商的TrustZone硬件隔离实现机制各异	如何保证不同TEE方案的安全性？
2、大多TEE与手机厂商更关注功能性	如何确保TEE自身安全性？
3、TA不断增加，带来新的安全隐患	如何更有效地将TEE与TA隔离开？
4、TEE无法应对物理攻击	如何提高TEE防御的层次？

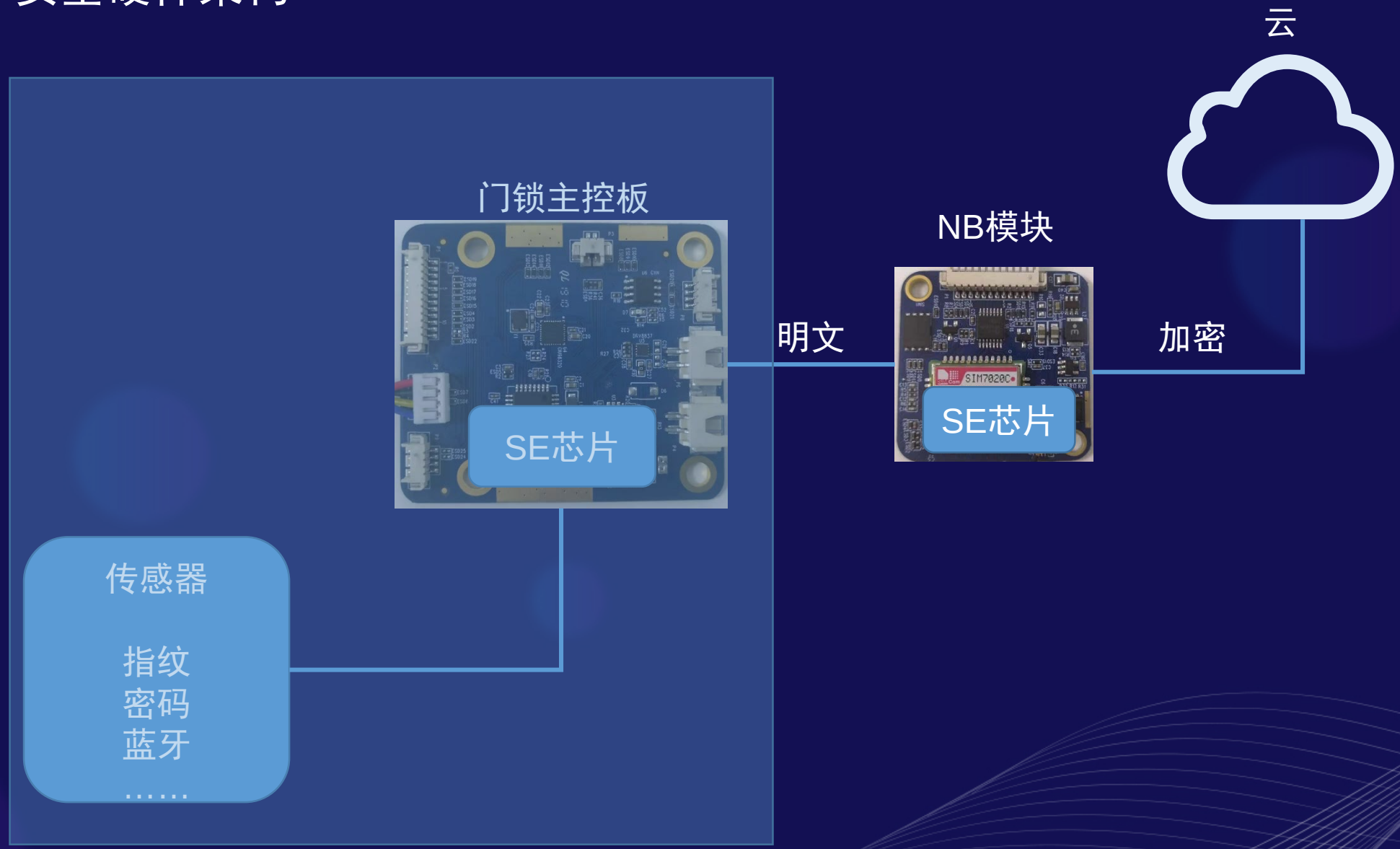
TEE不能绝对解决的问题

- 2019年4月，高通40款芯片的TEE出现漏洞，编号CVE-2018-11976，可导致攻击者获取存储在芯片TEE中的私人信息和密钥，波及数十亿安卓设备。

- 2018年三星手机的TEE环境出现漏洞，攻击者可执行任意代码并获取存储的密码信息。

一个悲观主义者的思考

安全硬件架构



谢谢！！！