物联网安全
Internet of Things Security

第六章 芯片安全

冀晓宇
浙江大学

芯片安全

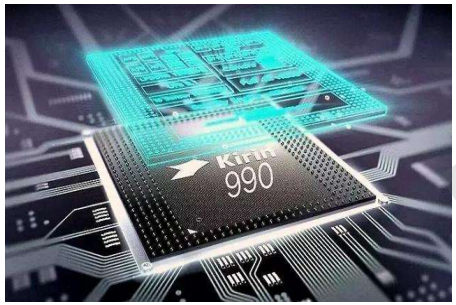
6.1 芯片概述

6.1.1 芯片的起源和发展

- **集成电路的发展历程**
  - 1947年，美国贝尔实验室发明晶体管。
  - 1958年，仙童公司与德州仪器分别发明了集成电路。
  - 1963年，提出CMOS技术。今天，95%以上的集成电路芯片都是基于CMOS工艺。
  - 1964年，Intel摩尔提出摩尔定律，预测晶体管**集成度**将会每18个月增加1倍。
  - 1974年，RCA公司推出第一个CMOS微处理器1802。
  - 1985年，Acorn公司做出了一台RISC指令集计算机，简称**ARM**。
  - 1988年，16M DRAM问世，1平方厘米大小的硅片上集成有3500万个晶体管，标志着进入超大规模集成电路（VLSI）阶段。
  - …
  - 2018年，台积电和三星已先后实现7nm制程的突破并量产。
  - 2019年，5G、AI芯片等不断发展……

6.1.2 什么是芯片

- **定义**
  - 芯片又称为集成电路(integrated circuit，IC)，是由**独立半导体**和**被动组件**集成到衬底或线路板所构成的小型化电路。
- **典型芯片**



麒麟990

A13

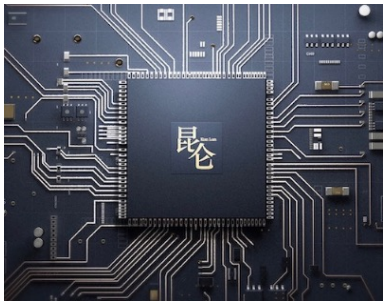
高通855+

- **规模分类**
  - 小型(SSI)、中型(MSI) 、大规模(LSI) 、超大规模(VLSI) 、极大规模(ULSI)
- **电路属性分类：**模拟、数字和混合集成电路

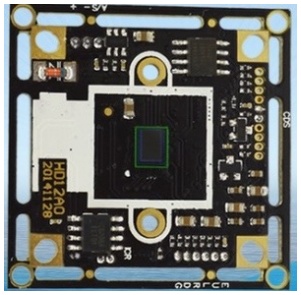
6.1.2 什么是芯片

- **物联网系统芯片的特点**
  - **专用性**：如ARM芯片专为嵌入式系统设计
  - **与物理世界交互**：传感、计算和控制类芯片

- **片上系统（SoC）成为主流**
  - SoC: system on chip，也称为系统级芯片，是一个有专用目标的集成电路，其中包含完整系统并有嵌入软件的全部内容
  - SoC的领域专用性：视频监控、深度学习计算、车联网等

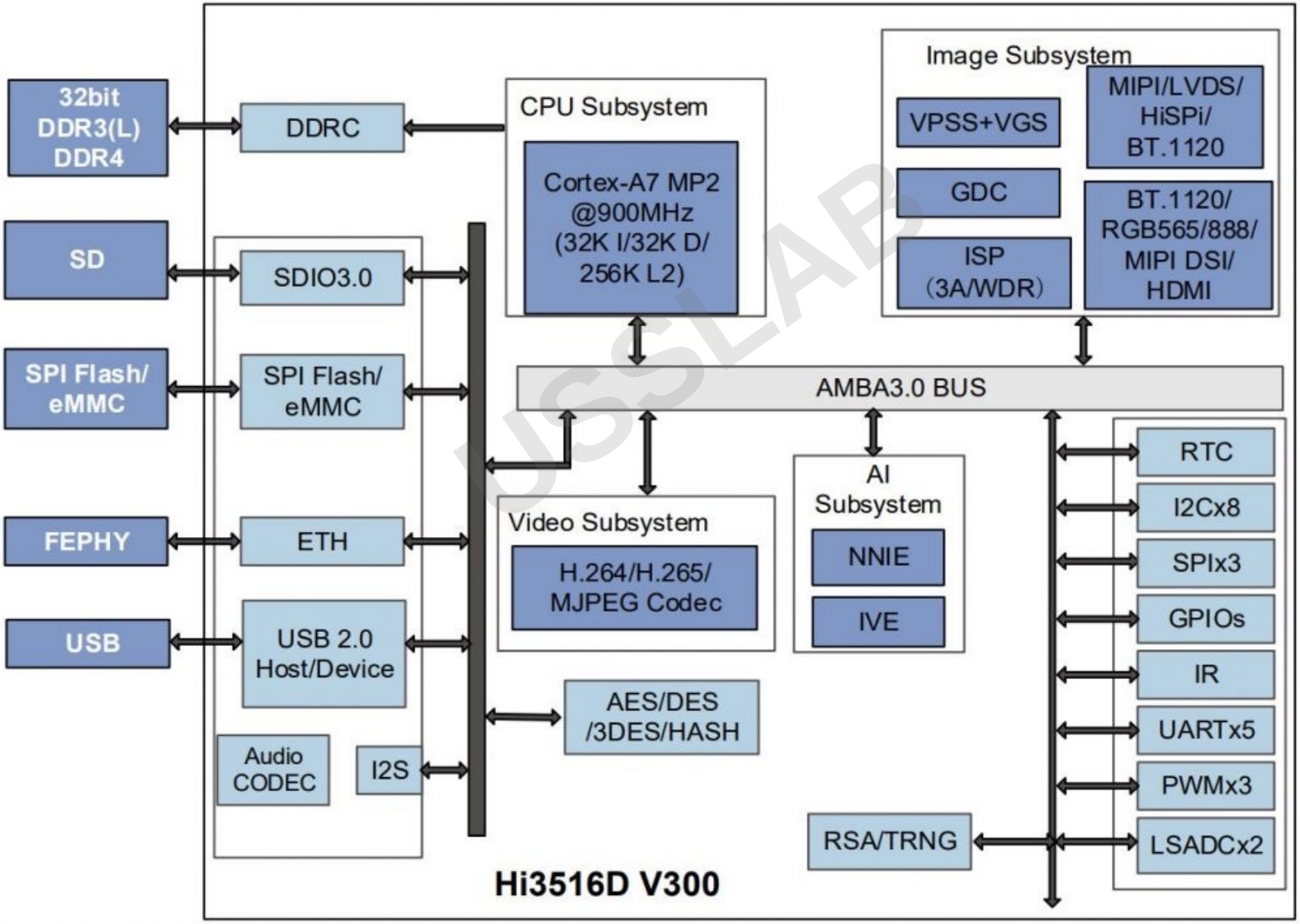


AI计算芯片



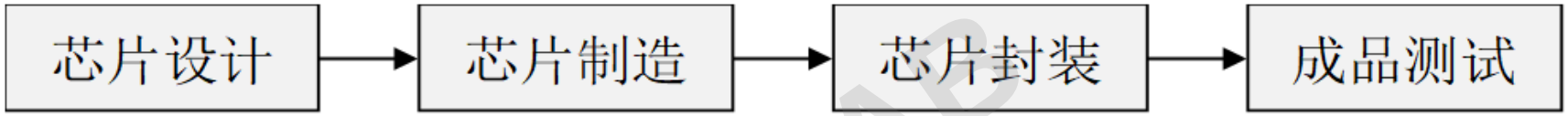
专用于摄像头的SoC芯片

案例：华为海思智能摄像头Hi3516D V300SoC芯片



6.1.2 什么是芯片

- **芯片生产链**



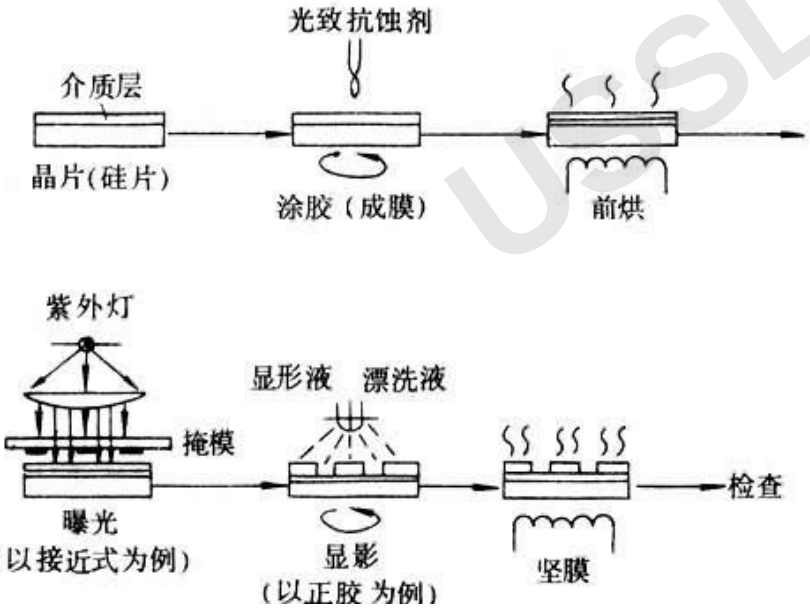
  - **芯片设计：**制定芯片规格，通过硬件描述语言（HDL）描绘电路。
  - **芯片制造：**依照设计好的电路，在硅晶圆上完成电路制造。
  - **芯片封装：**安装外壳以保护芯片，同时其引脚起到连接外部电路的作用。
- **最难的是设计**，其次是制造，最容易的是封装和测试。

6.1.2 什么是芯片

- **芯片设计技术**
  - IP核技术：intellectual property core（知识产权核），指**已验证的、可以重复使用的具有某种确切功能的集成电路设计模块**。
  - 是当前芯片设计技术的主流，著名的ARM公司以专业设计IP核在CPU领域占据一席之地。
  - 包括软IP、固IP、硬IP：
    - 软核通常是与工艺无关、具有寄存器传输级硬件描述语言描述的设计代码，**可以进行后续设计**；
    - 硬核是软核通过逻辑综合、布局、布线之后的一系列表征文件，具有**特定的工艺形式、物理实现方式**；
    - 固核则通常介于上面两者之间，它已经通过功能验证、时序分析等过程，设计人员可以以逻辑门级网表的形式获取。

6.1.2 什么是芯片

- **芯片制造技术**
  - 集成电路的**工艺特征尺寸越来越小**，芯片制造技术也不断发展。
  - 工艺要求：1皮米，10^{-14}米（头发丝的1/5000万）



光刻技术

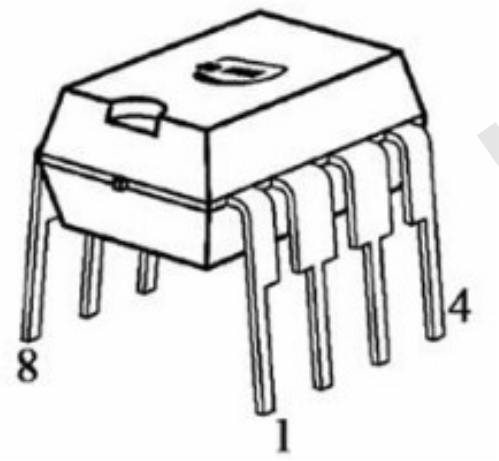
通过在光刻过程中不断改进曝光源的波长以适应更小的光刻尺寸。

ASML公司每台光刻机大约1亿美金

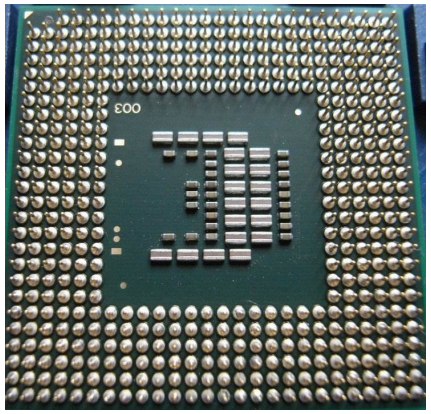
Q:芯片尺寸小型化和安全问题有何关系?

6.1.2 什么是芯片

- **芯片封装技术**
  - 安装半导体集成电路芯片用的外壳，起着安放、固定、密封、保护芯片和增强电热性能的作用
  - 通过引脚实现沟通芯片内部世界与外部电路的作用



19世纪70年代，双列直插
DIP(dual in line package)技术



19世纪90年代，球栅阵列封装
BGA(ball grid array)技术

物联网时代下的“芯机遇”——RISC-V

RISC-V

精简指令集（1980）

Reduced Instruction Set Computing

第五代精简指令集（2010）

We invented RISC-V



Krste Asanović　Yunsup Lee　Andrew Waterman

David Patterson

- 2017图灵奖得主
- 美国加州大学伯克利分校教授
- RISC发明者
- SiFive公司技术顾问

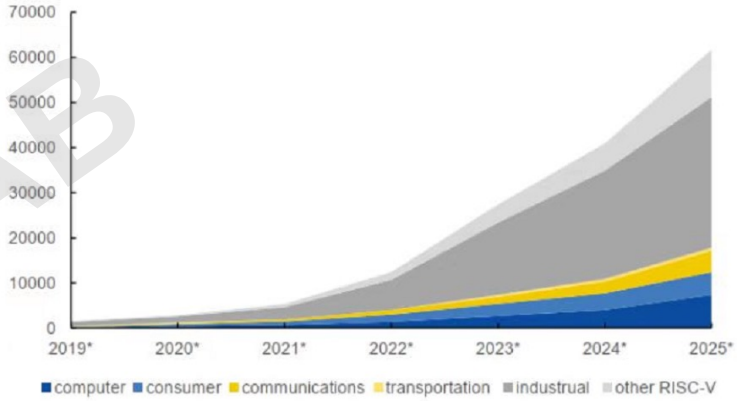
- 开源：任何人、任何目的、免费自由使用
- 架构简单：仅47条基础指令，用户可按需扩展（X86 1500条指令）
- 易移植，并可实现模块化设计，能耗低
- 规避美国贸易限制：2020年3月RISC-V基金会从美国迁移到瑞士

RISC-V背景

- **物联网时代带来了大量且碎片化的需求**

全球物联网与非物联网设备数（单位：亿）

RISC-V出货量预测（单位：百万）



- **三大指令架构商业模式**

微架构设计 / 指令集	① 开放免费的设计	② 需授权的设计	③ 封闭的设计	产品可选的设计（对应各指令集）
开放免费的指令集（RISC-V）	Berkeley的Rocket Chip/剑桥lowRISC/芯来科技蜂鸟E203	平头哥/SiFive/晶心科技Andes的RISC-V处理器核	Google和NVIDIA的自研RISC-V处理器	① ② ③
需授权的指令集（ARM）		ARM的处理器设计，如Cortex-A76等	基于ARM架构的Apple处理器	② ③
封闭的指令集（x86）			Intel和AMD的处理器	③

“在物联网时代，场景需求碎片化、差异化，低成本、定制化的RISC-V架构在物联网时代优势尽显。”
——2020年物联网RISC-V行业深度研究报告》

RISC-V或是国产芯片弯道超车的机会

倪光南
中国工程院院士

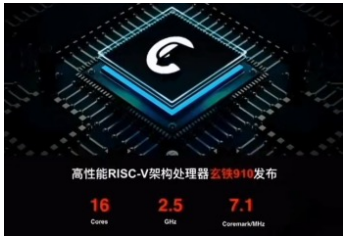
“RISC-V是国产芯片弯道超车的机会，未来将会和X86以及ARM架构，形成三足鼎立的形势！”

- **RISC-V 基金会最高级别会员大多数是中国企业**

- **国内企业对RICS-V的青睐**

华为Hi3731V110
(显示屏)

紫光展锐 春藤5882
(tws蓝牙耳机)



阿里巴巴平头哥
玄铁910

华米 黄山1号
(智能手表)

6.1.3 芯片的功能和分类

- **物联网专用芯片主要种类**

芯片类型	典型芯片
计算	CPU、DSP、FPGA、MCU
感知	MEMS传感器、指纹传感器、麦克风、摄像头
存储	DRAM、SDRAM、NDND、FLASH
通信	蓝牙芯片、Wi-Fi芯片、NB-IoT芯片、光纤芯片
供能	电源芯片、功率芯片
接口	USB芯片、以太网接口芯片、HDMI接口芯片

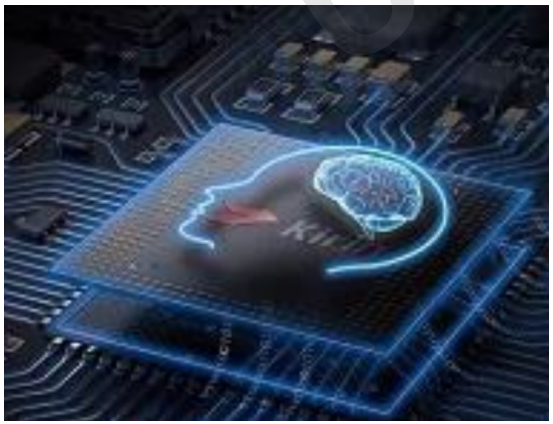
6.1.3 芯片的功能和分类——计算芯片

- **计算型芯片主要类别**
  - **微处理器**
    - 一般由算术逻辑单元、控制逻辑单元和控制存储单元组成
    - 根据不同用途：包括CPU、GPU(Graphic Processing Unit)、APU(Accelerated Processing Unit)、NPU(Neural Processing Unit)等，APU=CPU+GPU，NPU神经网络处理器，适合实时图像识别、语音处理，例如Apple M3芯片中的16核NPU。
  - **微控制器MCU**
    - 又称单片机，是把中央处理器、存储器、定时/计数器、各种输入输出接口等都集成在一块集成电路芯片上的微型计算机，在物联网中被广泛采用
  - **FPGA**
    - 现场可编程逻辑门阵列，是一种半定制电路，内部逻辑可反复更改
  - **DSP**
    - 一种专用的数字信号处理的微处理器，通常用于音视频信号处理

物联网专用芯片——AI计算芯片

- AI芯片是指能够运行人工智能算法的芯片
- AI芯片针对人工智能算法做了特殊加速设计的，现阶段人工智能算法一般以深度学习算法为主

- 代表厂商：英伟达、中科寒武纪、中星微、深鉴科技、灵汐科技、华为等。



华为麒麟 970 神经网络处理器 NPU

功能：

- 拍照场景识别
- 人体姿态识别
- 视频人像分割

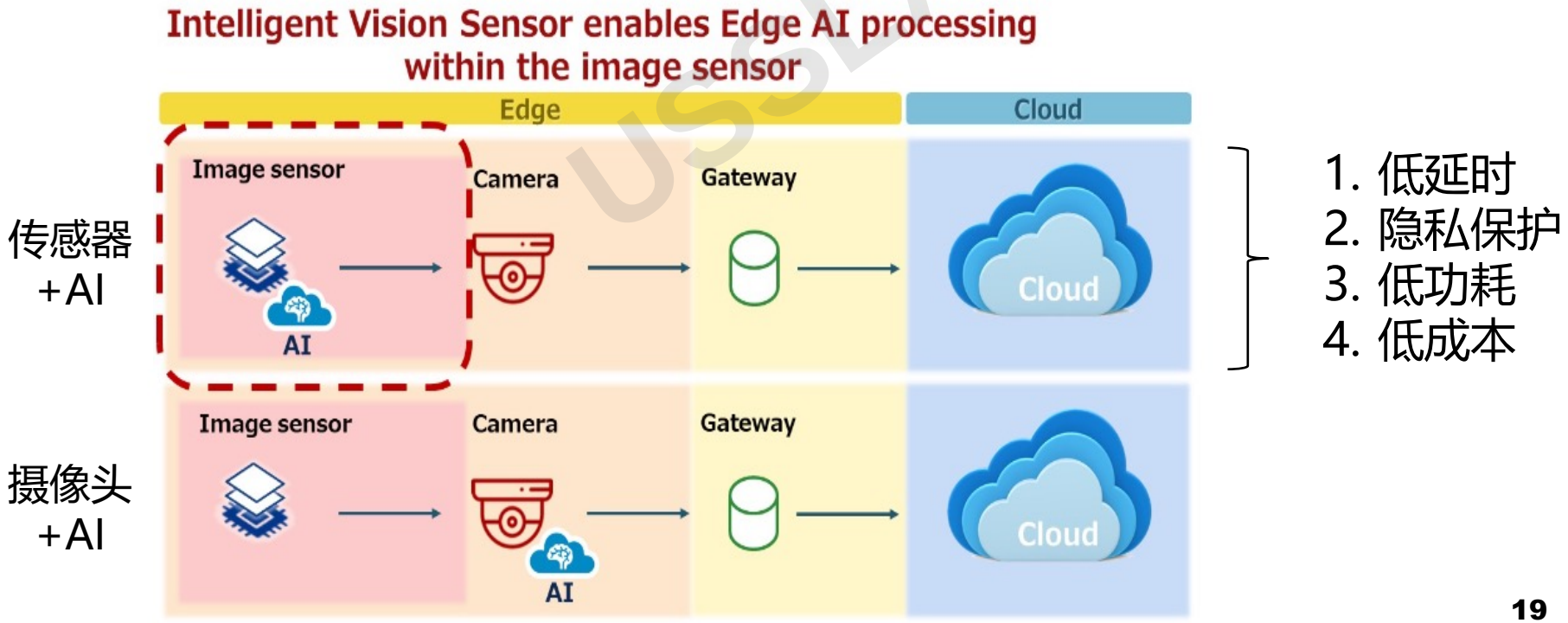
物联网专用芯片——传感器芯片

- 传感器技术历经了多年的发展
  - 第一代：**结构型传感器**，利用结构参量变化来感受和转化信号
  - 第二代：**固体型传感器**，这种传感器由半导体、电介质、磁性材料等固体元件构成

- 未来趋势——智能传感器
  - 具有信息采集、信息处理、信息交换、信息存储功能的多元件集成电路，是**集传感芯片、通信芯片、微处理器、驱动程序、软件算法等于一体的系统级产品**
    - 传感器端计算：将部分传感器数据运算能力封装在传感器芯片中，尤其是AI功能集成化，传感器愈发“智能化”
  - 智能传感器：如AI摄像头传感器、AI麦克风传感器等

传感器芯片集成AI

- 不输出图像信息而输出元数据（属于成像数据的语义信息），减少处理量；
- 不输出图像信息，可降低安全风险，保护隐私；
- 除拍摄图像，还可根据用户需求选择ISP、输出格式及ROI区域。

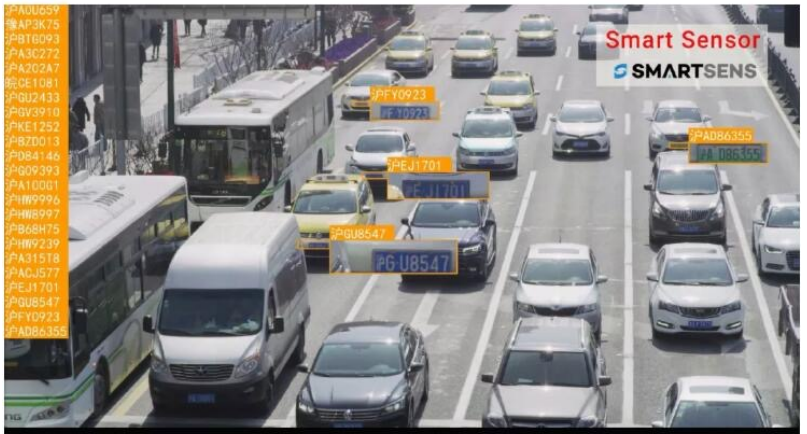


物联网专用芯片——传感器芯片举例

采用传统4K 分辨率sensor受到距离角度等因素限制，在人口密集的地方无法清晰抓拍到高分辨率的人脸细节，造成了误判率的提高

传统4K sensor车牌识别受到分辨率限制，只能识别较近距离的车牌

采用“SmartSensor”AI智能传感器芯片平台，可以清晰实时地捕捉整个画面中所有人脸信息，不放过任何一个表情细节



采用“SmartSensor”AI智能传感器芯片平台，可以超越4K限制清楚和高效地捕捉整个画面中的所有车牌信息

物联网专用芯片——安全芯片

- 安全芯片：作用相当于一个“保险柜”，将重要的**密码、人脸、指纹**等机密数据都存储在安全芯片中
- 密码数据：根据安全芯片的原理，密码数据只能输出，不能输入，加密和解密的运算在安全芯片内部完成，只是将结果输出到上层应用，避免了密码被读取破解的可能
- 代表厂商：国民技术，华大电子等
- 典型技术：TEE

HSC32I1超低功耗物联网安全芯片

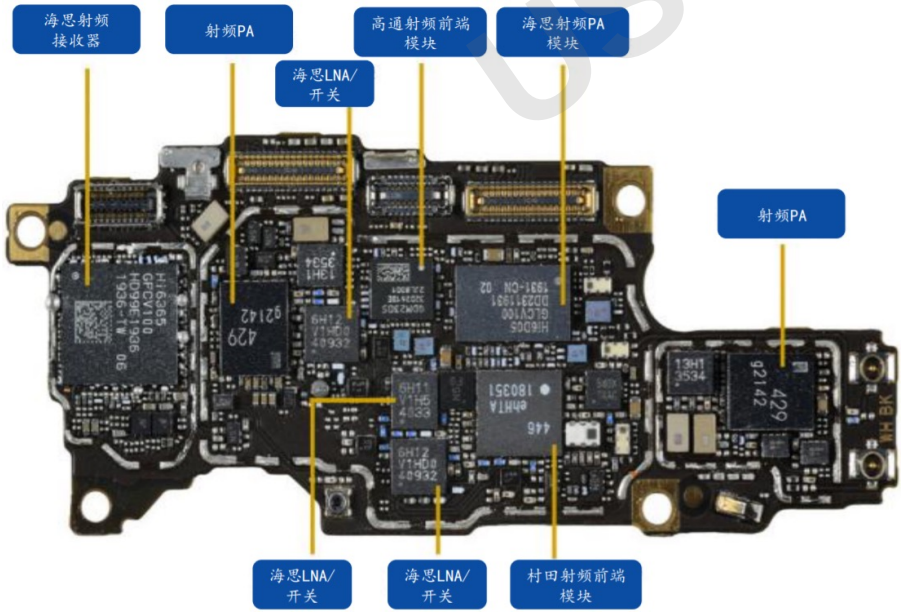
物联网专用芯片——移动支付芯片

- 物联网时代，越来越多的智能自助终端将出现在我们的工作生活当中，而移动支付的普及正是最大推手
- 目前国内移动支付方案主要是支持NFC功能的SIM卡/SD卡产品。移动支付芯片主要用于智能手机，因此移动支付芯片的需求主要来自于新增手机以及更新迭代
- 代表厂商：大唐微电子，中科达等

中科创达 TurboX™ S820 SOM 移动支付芯片

物联网专用芯片——通讯射频芯片

- 射频芯片：将无线电信号通信转换成一定的无线电信号波形，并通过天线谐振发送出去的芯片。
- 物联网专用射频芯片：NB-IoT、5G毫米波、蓝牙、低功耗蓝牙（LBE）等
- 代表厂商：新岸线，中兴微电子等



华为Mate30射频芯片

物联网专用芯片——身份识别芯片

- 身份识别利用指纹、声音、面部、虹膜和DNA等人体生物特征，以及签名的动作、等个人的行为特征完成认证
- 身份识别芯片主要完成对身份特征的信号采集、处理和识别，如指纹和人脸专用芯片
- 代表厂商：汇顶科技，同方微电子等

FPC1011F3电容指纹采集芯片

思考

- 芯片会存在什么样的安全问题?
- 这些安全问题来源于哪里?
- 国产自主可控芯片的重要性。

6.1.4 芯片的安全威胁

- **硬件木马安全事件**


论 见

警惕『硬件木马』来袭

软件层面的病毒易躲 硬件中的『木马』难防

早在2003年6月，美国国会报告中就明确指出了“硬件木马”对国家安全的威胁，引起白宫的极大关注。其后，美国国防部称，F-35联合攻击战斗机可能使用了数个“危险的芯片”。意识到潜在风险后，美国国防部高级研究计划局启动了一个为期3年的“集成电路可信工程计划”，该计划的研究可以给军方和生产敏感微电子设备的国防承包商提供确保其微电子系统“可信”的方法。

目前，“硬件木马”已成为某些国家打击对手的重要手段之一。1991年的海湾战争中，美军通过激活设置在打印机芯片中的“木马”，侵入伊拉克防空指挥系统，导致其在开战伊始就陷入瘫痪，丧失制空权。2007年，为扼杀叙利亚核计划，以色列进行“果园行动”空袭，非隐身战机深入战略纵深地带摧毁预定目标并全身而退。整个行动中，叙军先进的防空系统没有做出任何反应。有外媒报道称，根据美国国防部供应商匿名提供的情况，一个“欧洲芯片制造商”在叙军防空武器系统的微处理器中加入了可以远程访问的“毁灭开关”。

6.1.4 芯片的安全威胁

- **边信道攻击安全事件**

芬兰 Tampere 理工大学和古巴哈瓦那理工大学的研究人员报告了针对英特尔处理器的新边信道攻击，新的漏洞被称为 PortSmash，研究人员称它影响所有使用同时多线程(SMT)架构的处理器，他们在英特尔的私有 SMT 实现 Hyper-Threading (HT)上确认了这一漏洞。利用 SMT 的并行运行能力，攻击者可以在合法进程旁运行恶意进程，利用合法进程泄露的少量数据重组其处理的加密数据。研究人员在 GitHub 上公布了概念验证攻击代码。

英特尔CPU漏洞再度袭来：研究员发现全新边信道攻击方法BranchScope

FreeBuf

发布时间：18-03-28 23:07

近日，美国四所大学的一组学者发现了全新的边信道攻击方法，他们能够利用现代CPU中的推测执行功能来获取用户CPU数据，泄漏敏感数据和数据安全边界。这种边信道攻击方法与今年年初的 Meltdown 和 Specter 漏洞利效果相似，但研究人员这次利用的是CPU推测执行功能中的一个新片段。

作者最新文章

预告 | 黑镜调查：深渊背后的真相之「DDoS威胁与黑灰产业调查」报告

6.1.4 芯片的安全威胁

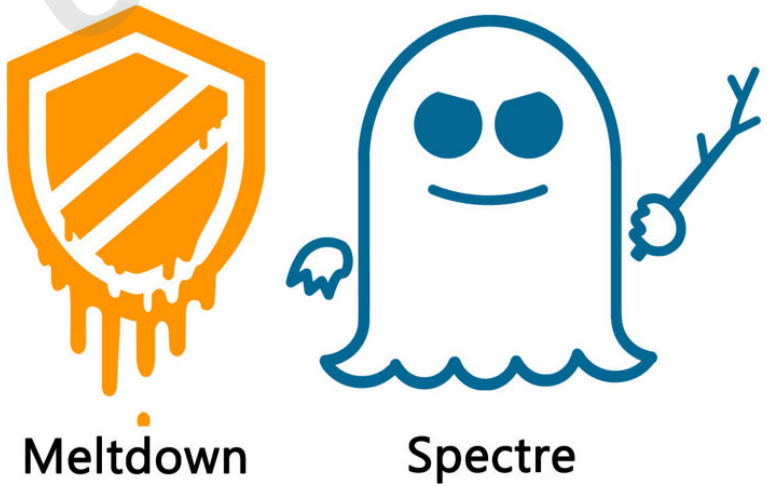
- **典型案例：Meltdown & Spectre**

Meltdown和Spectre两个严重的硬件错误，数十亿设备面临攻击风险

来源：Linux公社 作者：鱼鱼

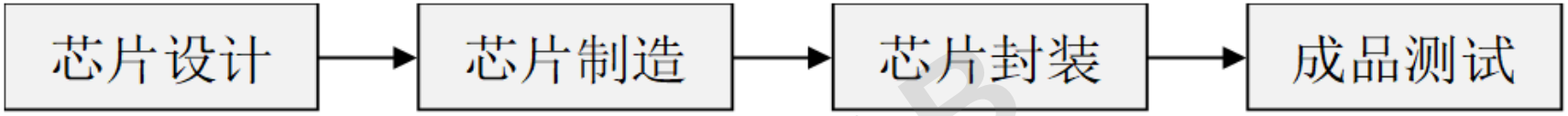
重大处理器漏洞Meltdown(熔毁)和Spectre(幽灵)应急通报

2018-01-04 23:59



6.1.4 芯片的安全威胁

- **为什么芯片会存在安全问题**



- **本质原因**：芯片从需求产生到成品上市的过程，涉及众多高精尖技术，目前还没有企业或政府部门能够独自打造一条**完全可信的芯片生产供应链**

- **现实原因**：生产商为了减少成本、缩短生产周期而选择外包，并引用第三方EDA工具和IP核，导致**芯片设计和制造分离**，进一步增加了芯片的安全隐患

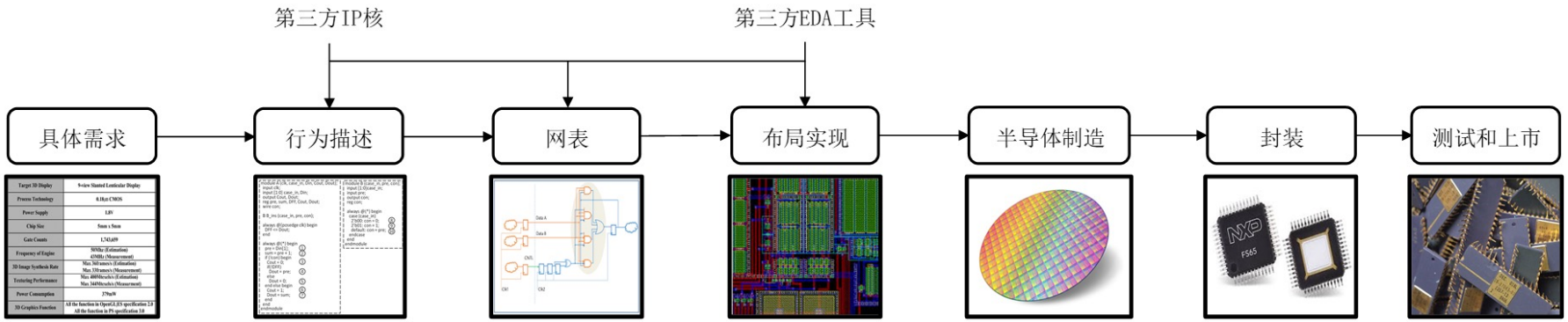
6.2 芯片的漏洞和威胁

6.2.1 芯片漏洞和安全威胁分类

- 安全威胁一：硬件木马
- 安全威胁二：边信道攻击
- 安全威胁三：存储器读取攻击

6.2.1 硬件木马

- **定义**：芯片从设计到制造的过程中，被蓄意植入或更改的特殊电路模块、或者是设计者无意留下的设计缺陷
- 在特定情况下，硬件木马会被恶意代码启动触发，达到干扰正常工作、窃取数据、破坏系统等目的，也称为“硬件后门”
- 硬件木马来源
  - 芯片供应链每个环节或单位都有可能植入芯片后门或硬件木马
  - 第三方厂商参与增加了安全的风险



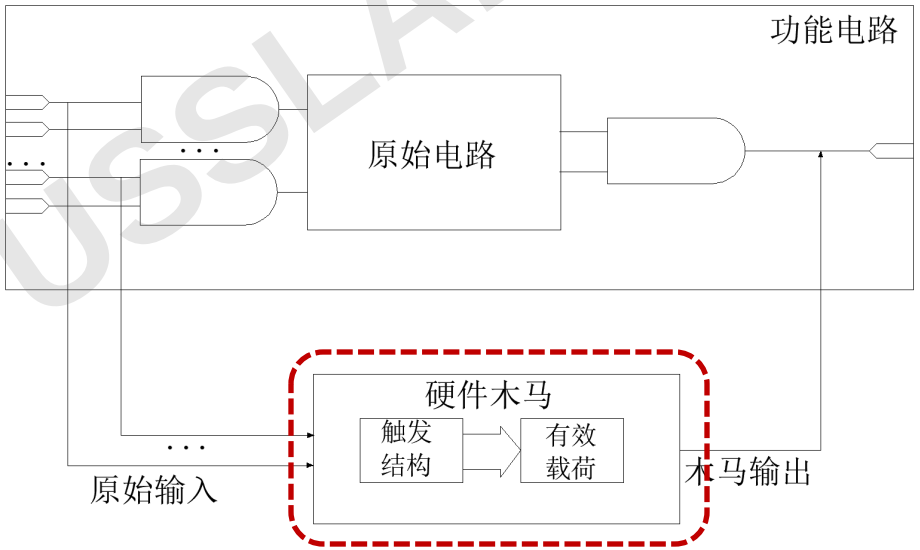
6.2.1 硬件木马

- **芯片攻击者模型**

攻击者模型	描述	第三方IP供应商	SoC开发商	半导体制造公司
A	不可信的第三方IP供应商	不可信	可信	可信
B	不可信的半导体制造公司	可信	可信	不可信
C	不可信的EDA工具或员工	可信	不可信	可信
D	最新的商业化元件	不可信	不可信	不可信
E	不可信的设计	不可信	不可信	可信
F	IP可信，但SoC开发商不可信	可信	不可信	不可信

6.2.1 硬件木马

- **硬件木马组成**
  - **触发器/结构**：用来检测电路中的各种信号和/或一系列事件来激活载荷
  - **有效载荷**：硬件木马的攻击单元，一旦触发器检测到预期的信号或者事件，就激活有效载荷以执行恶意行为

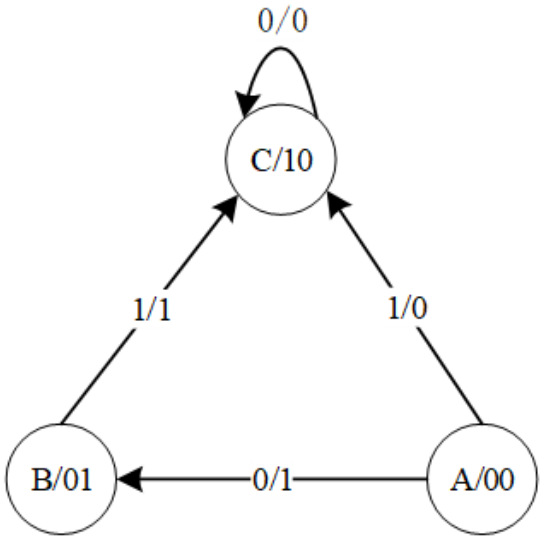


- **硬件木马状态**
  - **不活跃**：当有效载荷处于不活跃状态时，芯片的表现和没有木马一样
  - **激活**：触发器极少被激活，有效载荷在大多数时间内保持不活跃状态

6.2.1 硬件木马

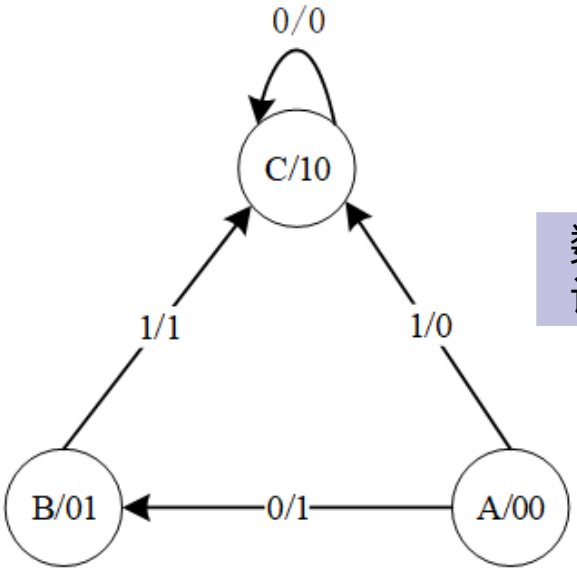
硬件木马设计示例

- 硬件电路设计用有限状态机（FSM）表示
- 考虑要设计右图所示三态有限状态机，可见用户没有访问状态A的权限，**故状态A可被认为安全**，因此可以用于存储隐私信息
- 注意：当系统状态为B，输入x=0时，并没有指定下一个状态和输出，这种情况被称为无关转换

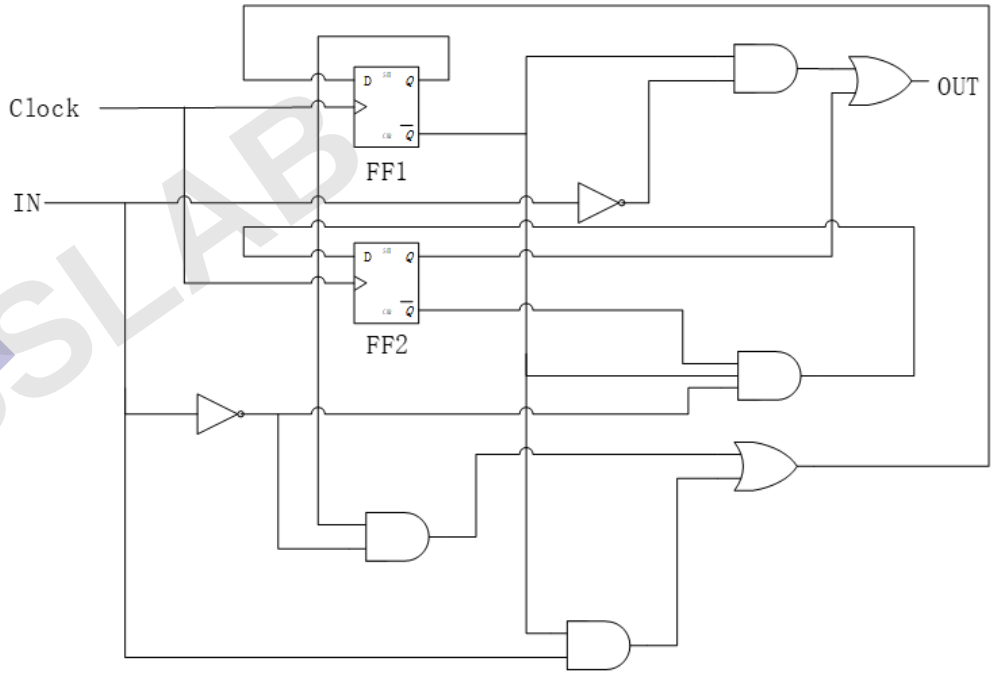


6.2.1 硬件木马

- **硬件木马设计示例**



数字电路
设计实现

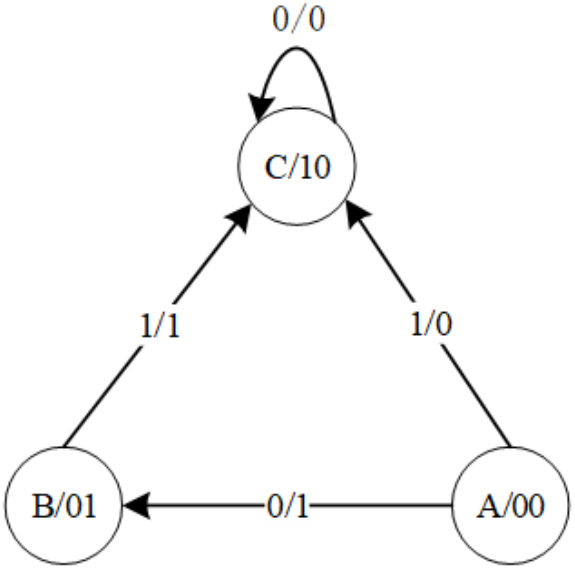


左图FSM的电路实现为上图，记为FSM’

注意：当处于状态B（即FF1=0，FF2=1），如果输入x=0，FF1输出不变，FF2输出将变为0，此时系统状态到A，**出现理论设计下不应出现的状况！**

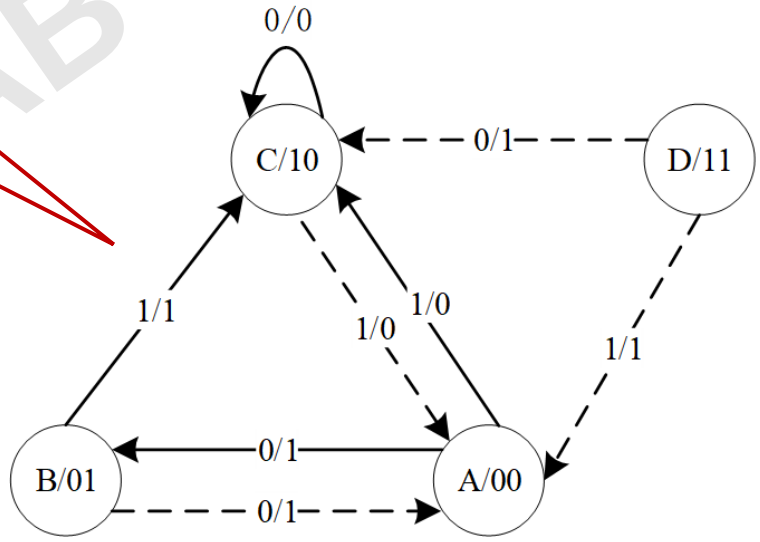
6.2.1 硬件木马

- **硬件木马设计示例**
  - 对前述电路FSM’进行功能分析，得到右图FSM’的状态转换图



FSM

状态增多
转换路径增多

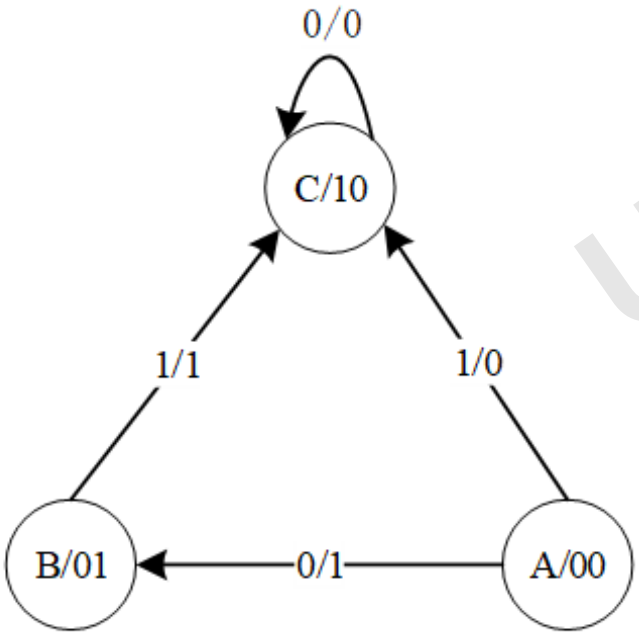


FSM’

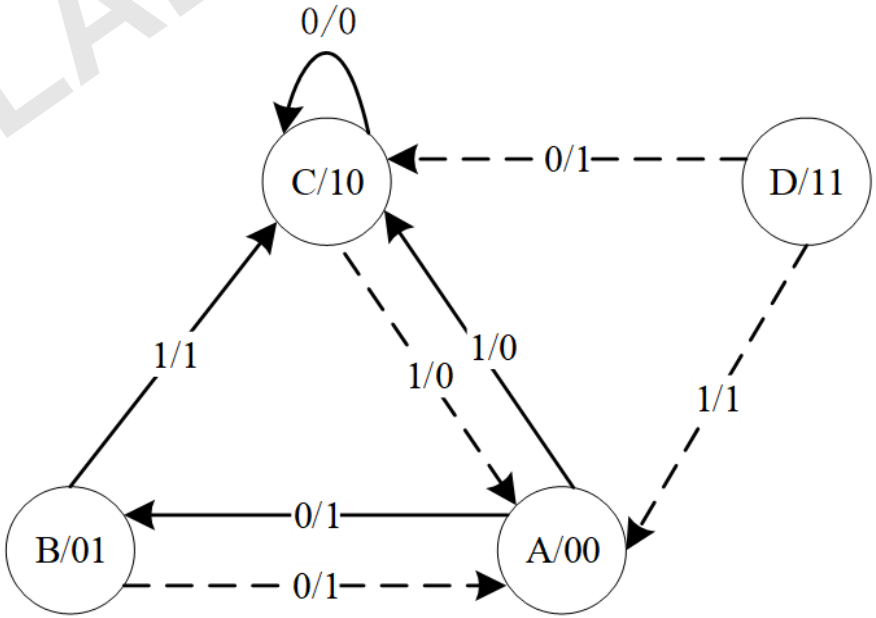
理论设计FSM中A为安全状态，实际实现电路FSM’中A状态有多条路径可达，并增加了状态D，该设计违反了对状态A的访问控制！

6.2.1 硬件木马

- **硬件木马设计示例**
  - **攻击场景**：为便于描述，定义**R(u)**表示所有从状态u能到达的状态，定义**S(u)**表示所有能够到达状态u的状态。



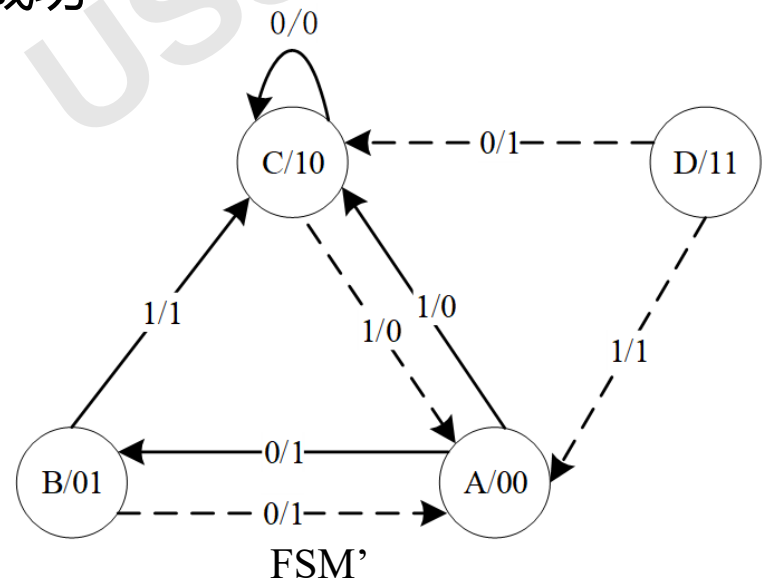
FSM



FSM'

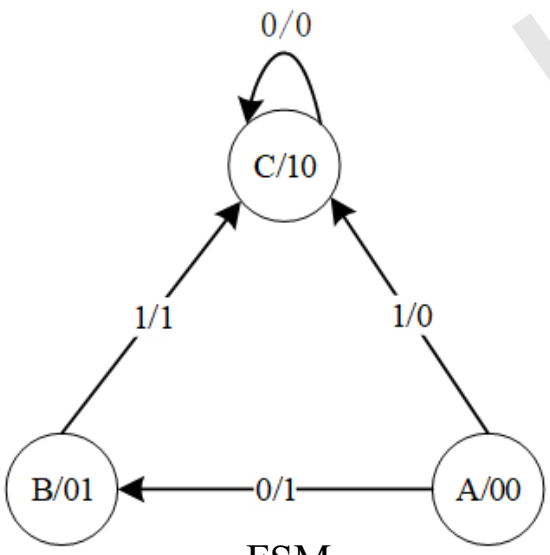
6.2.1 硬件木马

- **攻击场景1：攻击者只能访问FSM'的逻辑实现**，攻击目标是访问原始FSM中规定的无法访问的状态（如A）
- **攻击方法：**攻击者可尝试随机输入序列，并期望能够进入不应该到达的状态（即根据设计规范无法到达的状态）
- **示例：**攻击者初始状态为B，尝试输入1，则达到状态C后再通过输入1进入状态A，攻击成功

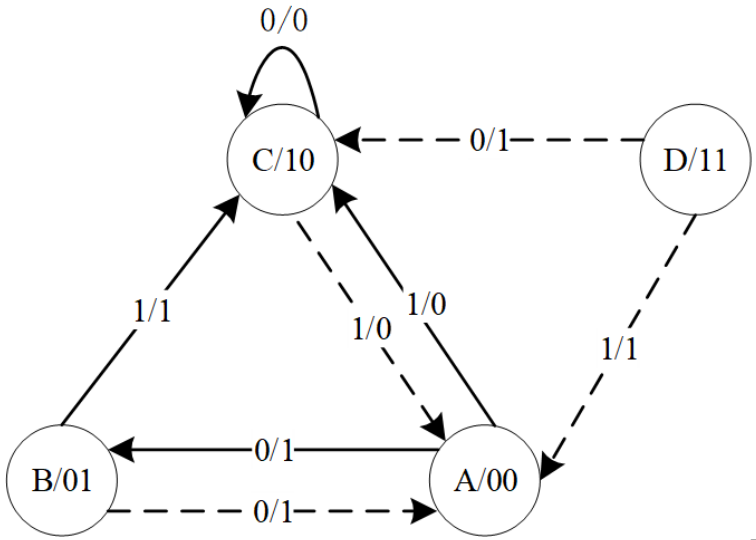


6.2.1 硬件木马

- **攻击场景2：攻击者获取了原始电路规范FSM**，并希望建立一条路径以达到某个不可达状态而不被检测到，即从状态u∉S(v)访问状态v
- **攻击方法：**检查FSM规范中，是否存在任何从u状态及R(u)集合中的状态开始的无关状态转换，也可利用逻辑综合工具引入新状态，并将新状态连接到状态v来获得对v的未授权访问
- **示例：**攻击者可从D状态对A进行访问



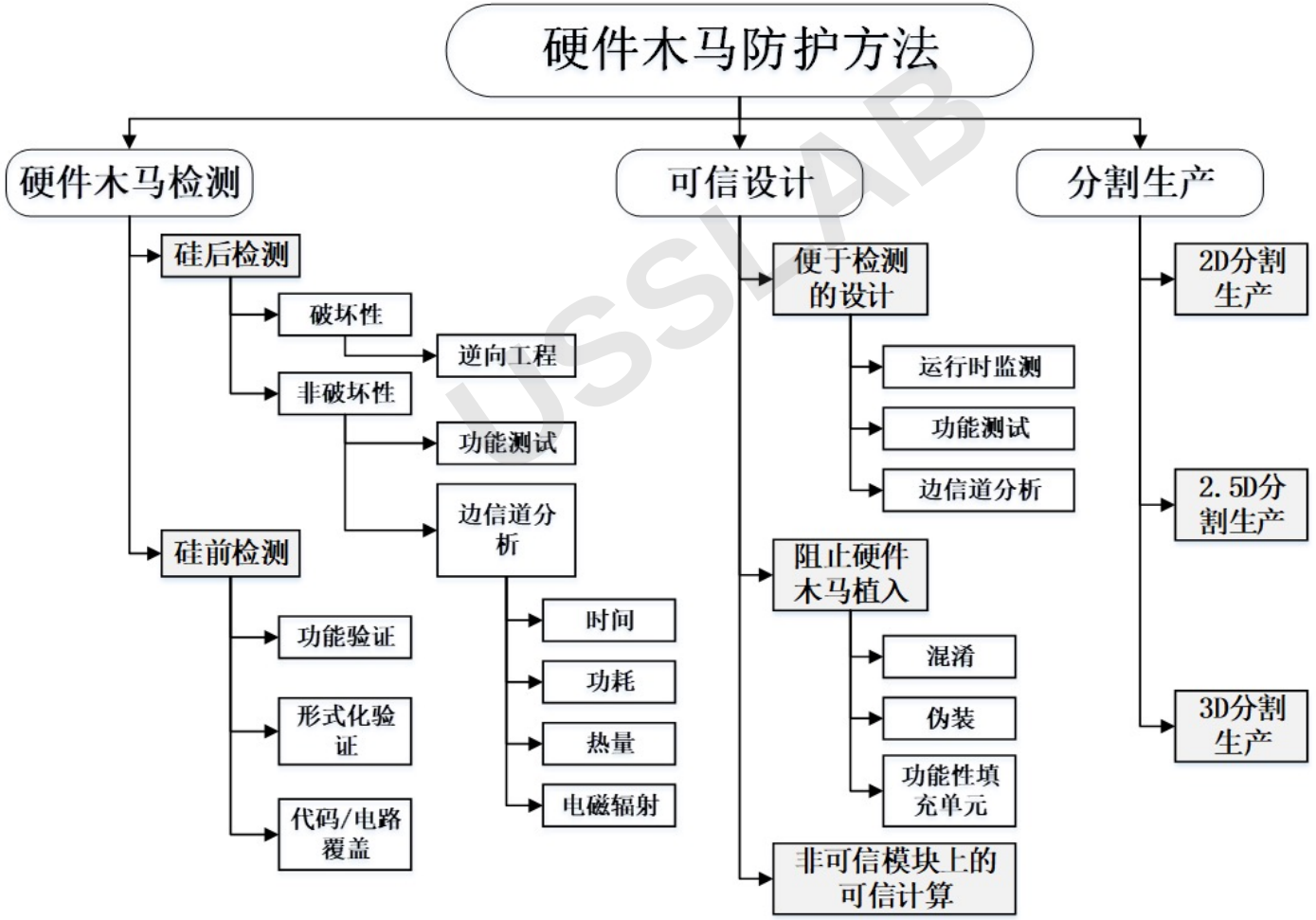
FSM



FSM'

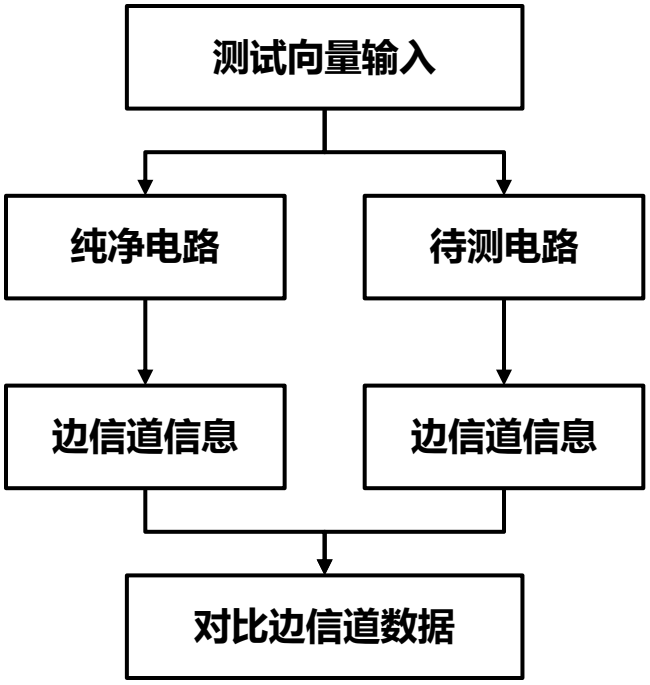
6.2.1 硬件木马

- **硬件木马防护方法分类**



6.2.1 硬件木马

- **硬件木马防护方法——木马检测**
  - **硅前检测**：在**设计阶段**检测硬件木马，分为功能验证，代码/结构分析和形式验证
  - **硅后检测**：在**制造阶段**之后检测硬件木马，分为**破坏性和非破坏性方法**
    - **破坏性方法**：使用破坏性逆向工程技术来拆解芯片并获取每层的图像，以便重建信任设计验证最终产品
    - **非破坏性方法**：功能测试或边信道信号分析（如右图所示）来验证来自不受信任的代工厂制造的芯片



6.2.1 硬件木马

- **硬件木马防护方法——可信设计**
  - **逻辑混淆：**在原始设计中插入内置锁定机制来隐藏设计的真正功能和实现
  - **伪装：**如布局级混淆技术，通过在伪装逻辑门内的层之间添加虚拟接触和伪造连接，为不同的门创建难以区分的布局，阻止攻击者提取电路的正确门级网表
  - **功能填充单元：**在布局设计期间用功能性填充单元填充所有空白空间，形成自检电路，实现芯片“水印”、“签名”效果
  - **不可信组件上的可信计算：**采用分布式软件调度协议来实现多核处理器中可容忍木马激活的可信计算系统

6.2.1 硬件木马

- **硬件木马防护方法——分割生产**
  - 分割生产将设计划分为前端线（FEOL）和后端线（BEOL），以供不同的制造厂制造。
  - 不受信任的代工厂执行FEOL制造，可靠的代工厂进行BEOL制造
  - 不受信任的代工厂无法访问BEOL中的层，因此无法找到硬件木马电路的插入位置。

6.2.2 芯片的边信道攻击 - side channel attack

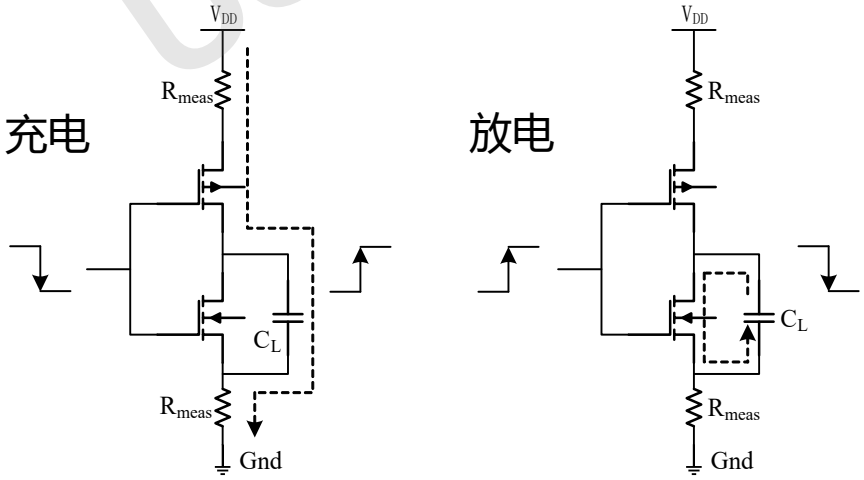
- **边信道攻击：**通过**时间信息、功耗、电磁泄露、声音等信号**在无权限条件下推测重要信息如密钥的一种攻击方式
  - **物理边信道**
    - 利用计算任务在芯片上执行时观测到的物理信号进行攻击，如**能量消耗、电场信号、声音信号**等
  - **缓存边信道**
    - 利用缓存是信息作为边信道，可分为时间驱动和访问驱动的缓存边信道攻击
  - **时间边信道**
    - 利用计算机**执行不同逻辑操作耗费的时间不同**，获取对攻击者有价值的信息

6.2.2 芯片的边信道攻击

- **物理边信道攻击——功耗边信道**
  - **原理**：CMOS芯片的功耗反映了其内部数据（指令）变化，动态功耗决定了芯片内部数据运算与功耗之间的关系，可以表示为：

$$p_{dyn} = C_L V_{DD}^2 P_{0\to1} f$$

  - 其中C_L为负载电容，$P_{0\to1}$是门电路0→1转换的概率，即数据计算造成的门电路翻转。f是芯片的工作频率，V_{DD}是芯片的工作电压。



Francois-Xavier Standaert, Introduction to Side-Channel Attacks

6.2.2 芯片的边信道攻击

- **物理边信道攻击——电磁辐射边信道**
  - 原理：芯片内部数据计算可以体现在芯片的电磁辐射信号，电磁辐射泄露通常可用Biot-Savart定律表示：

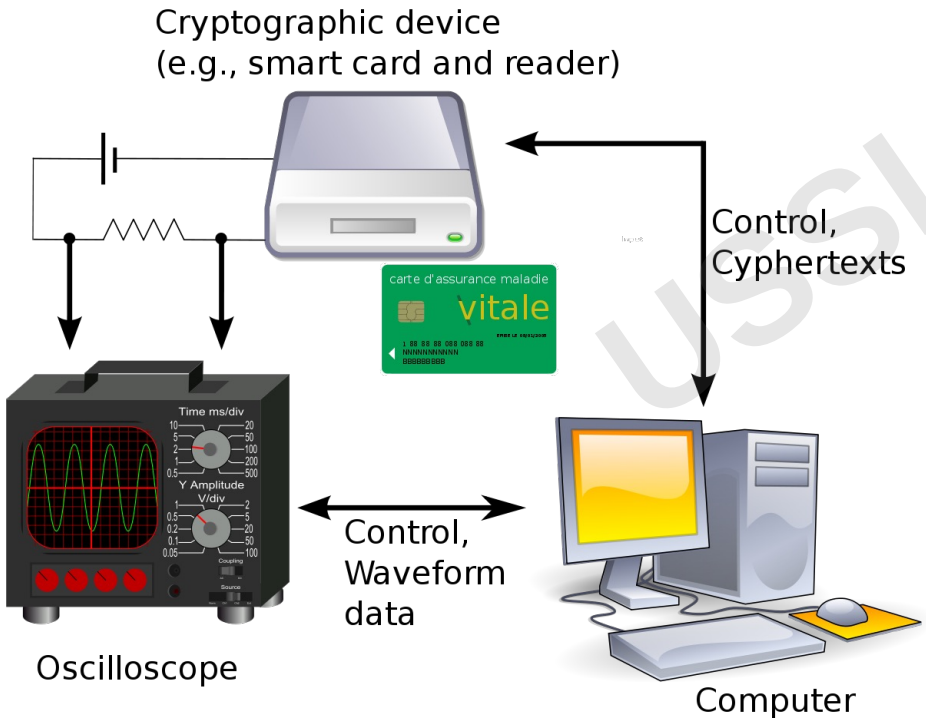
$$\mathrm{d}B = \frac{\mu I dl \times r'}{4\pi r^2}$$

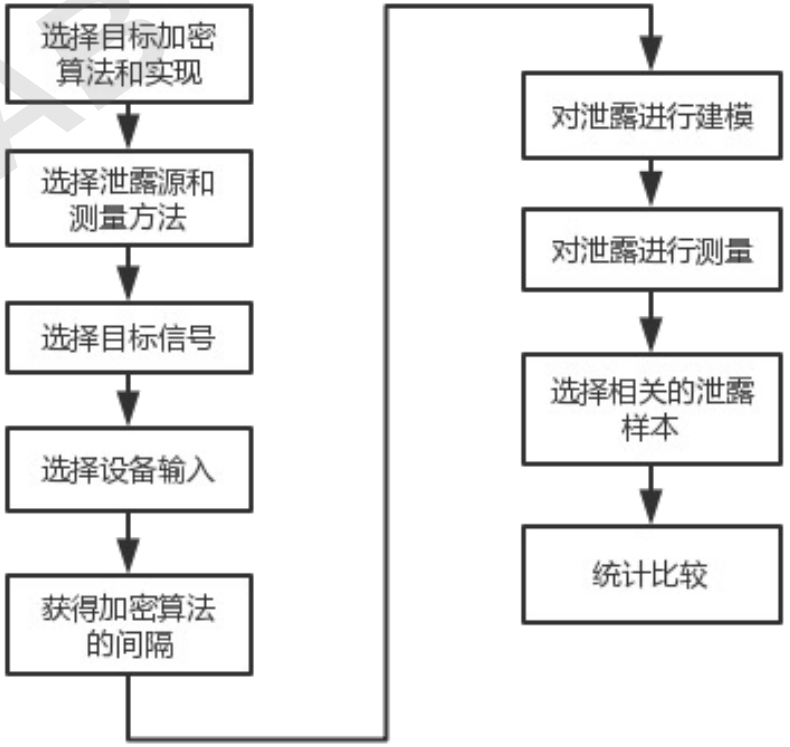
  - 其中μ是磁导率，I是无限长导体dl 段的电流，r'是电流元到场点的单位向量，r是电流元到场点的距离。

- CMOS芯片产生的**电磁辐射**信号和**计算过程强相关**

6.2.2 芯片的边信道攻击

- 物理边信道案例1——基于功耗边信道的DES密钥破解



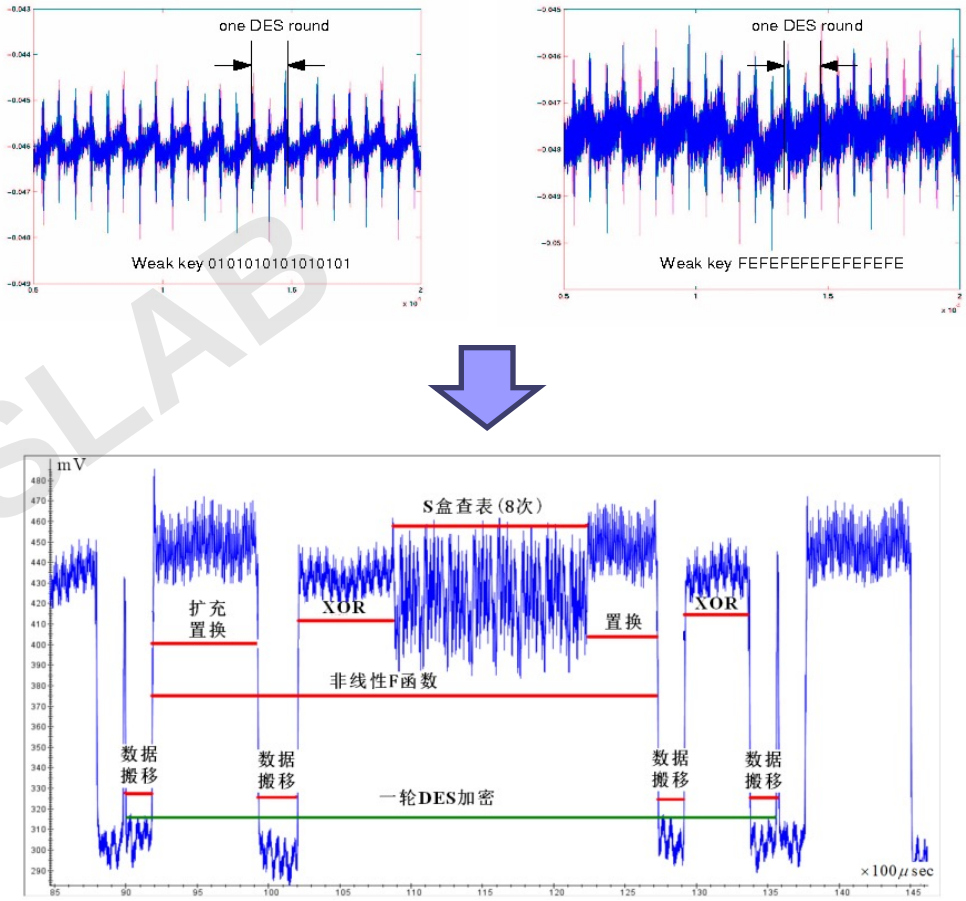


https://www.youtube.com/watch?v=GPwNFrpd1KU

P. Kocher, J. Jaffe, B. Jun, Differential Power Analysis, technical report, 1998; later published in Advances in Cryptology – Crypto 99 Proceedings, Lecture Notes in Computer Science Vol. 1666, M. Wiener, ed., Springer-Verlag, 1999.

6.2.2 芯片的边信道攻击

- **原理**：加密算法处理中时间与密钥信息的相关性构成了时间分析的物理基础
- **攻击步骤**：测量芯片功耗，将加密过程中**电路能量消耗与操作数关联**，用统计方法来获取密钥信息
- 统计学方法使得在某计算时刻芯片上与数据相关的功耗值得以累计，从而由功耗推导出数据值



P. Kocher, J. Jaffe, B. Jun, Differential Power Analysis, technical report, 1998; later published in Advances in Cryptology – Crypto 99 Proceedings, Lecture Notes in Computer Science Vol. 1666, M. Wiener, ed., Springer-Verlag, 1999.

More information on MIT open course: https://www.youtube.com/watch?v=MT7X17ZRo1U

RSA Power Analysis Side-Channel Attack

https://www.youtube.com/watch?v=bFfyROX7V0s

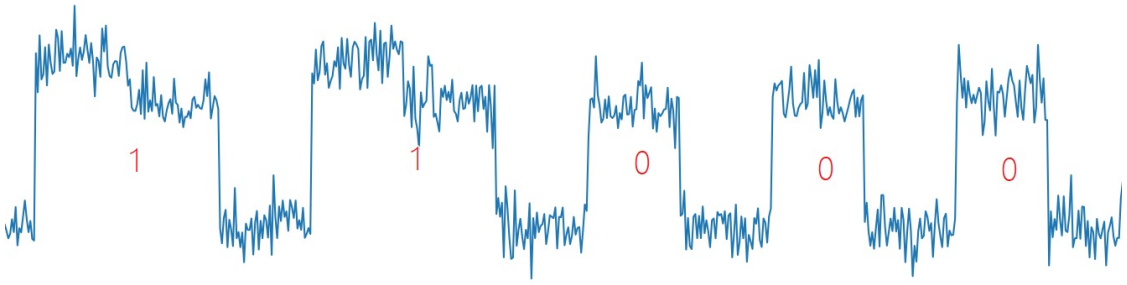
原理分析

- 芯片计算功耗与正在执行的操作存在相关性
- 理论基础：RSA采用快速幂取余算法

1. b 为偶数时，$a^b \ mod \ c = \left(a^{2^{b/2}}\right) \ mod \ c$ ⇒ **无乘法运算 功耗对应为 ‘0’**

2. b 为奇数时，$a^b \ mod \ c = \left(a^{2^{\lfloor b/2 \rfloor}} \times a\right) \ mod \ c$ ⇒ **多一步乘法运算 功耗对应为 ‘1’**

- 快速幂计算过程中会逐位判断指数（密钥）的取值，并会采取不同的操作，所以可从能量迹中还原出密钥的取值



RSA密钥 “...11000...”

原理分析

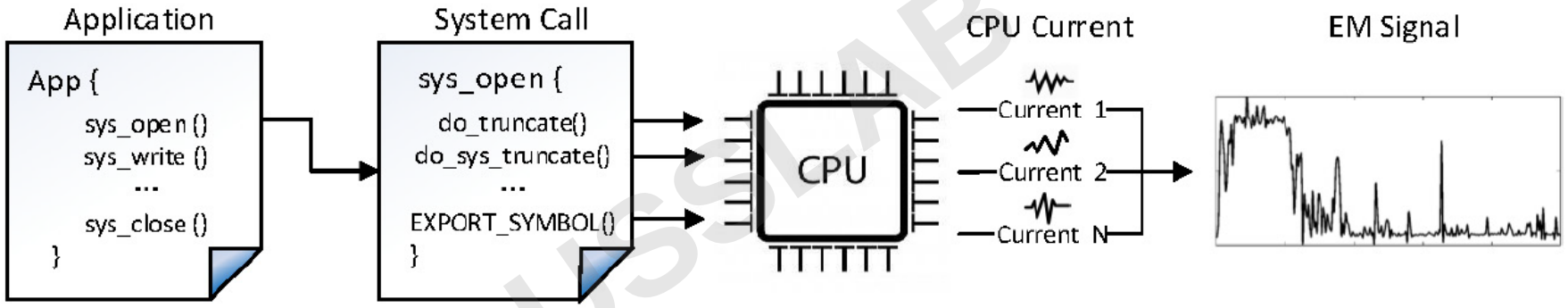
- a^b mod c 函数PowerMod()实现方法

1. b 为偶数时，$a^b \ mod \ c = \left(a^{2^{b/2}}\right) \ mod \ c$
2. b 为奇数时，$a^b \ mod \ c = \left(a^{2^{\lfloor b/2 \rfloor}} \times a\right) \ mod \ c$

```
int PowerMod(int a, int b, int c)
{
        int ans = 1;
        a = a % c;
        while(b>0) {
                if(b % 2 == 1) // 当b为奇数时会多执行下面的指令
                        ans = (ans * a) % c;
                b = b/2;
                a = (a * a) % c;
        }
        return ans;
}
```

6.2.2 芯片的边信道攻击

- **物理边信道攻击案例2——基于电磁的应用推测攻击**
- 案例：基于电磁边信道推断笔记本电脑上运行的应用程序



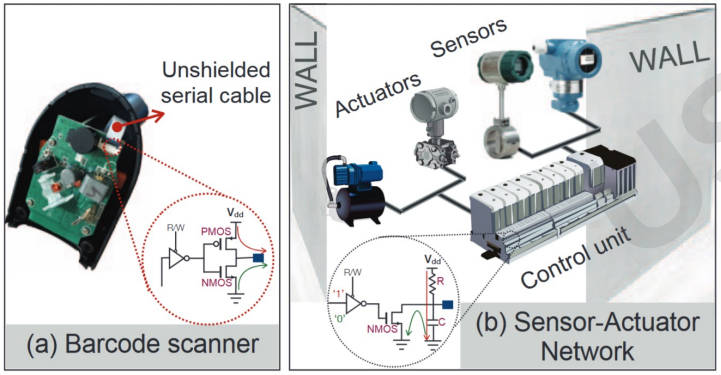
Yushi Cheng, Xiaoyu Ji, etc.. MagAttack: Guessing Application Launching and Operation via Smartphone. ACM Asia Conference on Computer and Communications Security (AsiaCCS), 2019.

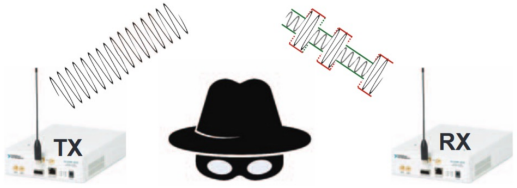
6.2.2 边信道攻击 -主动注入式

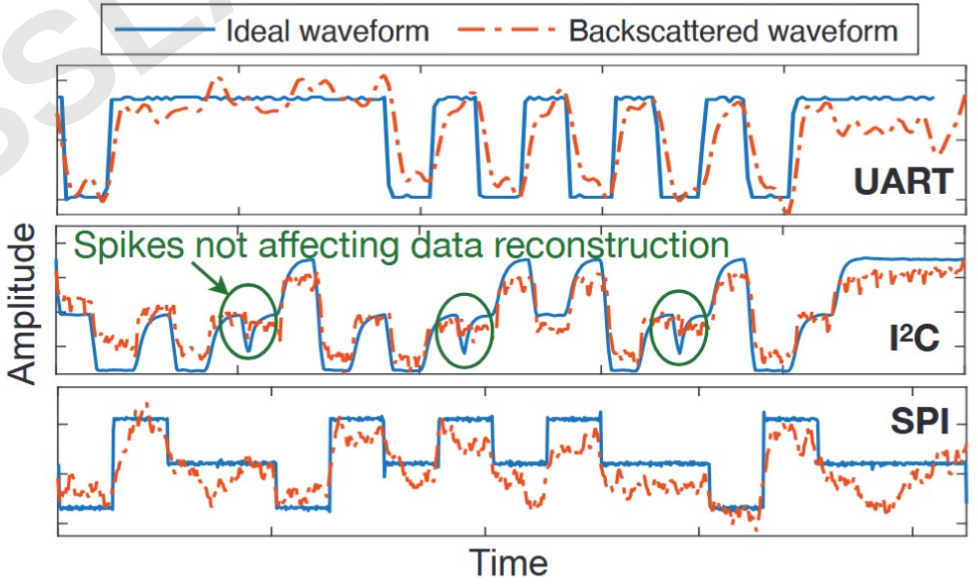
- **核心思想**:攻击者**主动发射探测载波**(RF/微波/激光/声波)，目标状态对回波进行被动调制，攻击者再从**反射/后向散射信号**中恢复信息
- **与传统被动式边信道的区别**
  - **被动侧信道：**目标自然泄露，SNR受限
  - **主动侧信道：**范式转变，可控激励+远距非接触
- **步骤**
  - **注入连续波信号作为载波**
  - **目标信号调制载波并反射（振幅、相位、频率）**
  - **接收反射信号并解调得到敏感信息**

6.2.2 主动注入式边信道攻击

- **攻击案例1——基于串口阻抗变化的RF泄露攻击**
  - 在 UART、SPI、I2C 等串行接口中，逻辑 0 和 1 信号传输会改变 I/O 端口的等效阻抗。
  - 射频连续波照射线缆后，端口阻抗变化会调制反射信号的振幅，形成调幅调制效果，攻击者可通过包络检波恢复数字比特流。



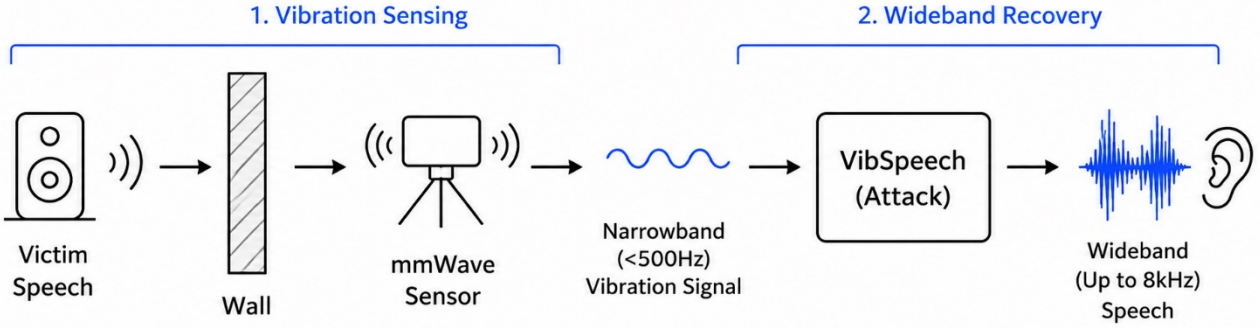




Pu, Lina, et al. "Your cable, my antenna: Eavesdropping serial communication via backscatter signals." 2025 IEEE Symposium on Security and Privacy (SP). IEEE, 2025.

6.2.2 主动注入式边信道攻击

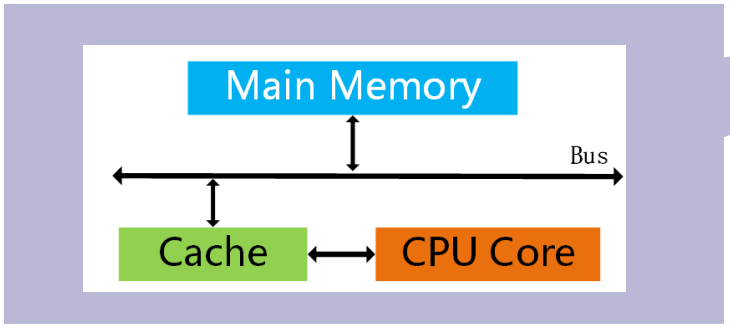
- **攻击案例2——基于声波机械振动的毫米波窃听攻击**
  - 声音会诱发玻璃、灯泡、笔记本外壳、水面等物体发生微米级甚至皮米级振动。
  - 毫米波照射这些表面后，反射路径长度变化会造成相位偏移，从而恢复语音

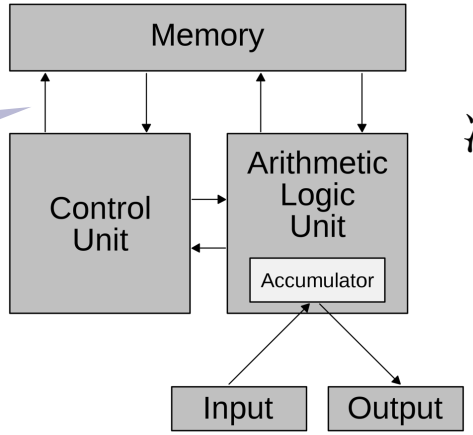


Wang, Chao, et al. "{VibSpeech}: Exploring practical wideband eavesdropping via bandlimited signal of vibration-based side channel." 33rd USENIX security symposium (USENIX Security 24). 2024.

6.2.2 芯片的边信道攻击——缓存边信道

- **缓存**：Cache，即中央处理单元高速缓冲存储器，容量比较小但速度比主内存更快
- **缓存的作用**：CPU发出内存访问请求时，会先查看缓存内是否有请求数据
  - 如果存在，即“命中”，直接返回该数据
  - 如果不存在，即“失效”，此时要先把内存中的相应数据调入缓存中，可以使得以后对整块数据的读取都从缓存中进行，从而加速数据访问
- **边信道原因**：程序运行时对内存访问呈现**空间局部性**和**时间局部性**





冯诺依曼架构

6.2.2 芯片的边信道攻击

- **缓存边信道攻击**：CPU多核之间缓存数据共享，而缓存命中和失效对应**响应时间有差别**，攻击者可以通过访问时间的差异，推测缓存中的信息，从而获得隐私信息

- **典型缓存边信道攻击“Flush-Reload”**
  - **前提**：攻击者和目标程序共享物理内存
  - **步骤一**：攻击者反复利用CPU指令把内存的某个地址清除出CPU缓存（Flush阶段）
  - **步骤二**：攻击者在一定的时间间隔后，读取这个地址上的内存数据并且测量读取的时间（Reload阶段）
  - **结果分析**：如果某个地址的读取时间明显小于其他，说明目标程序访问过该地址

6.2.2 芯片的边信道攻击

- 典型缓存边信道攻击：Meltdown和Spectre漏洞

6.2.2 芯片的缓存边信道攻击

- **预测执行：** Speculative execution，计算机利用空转时间提前执行一些将来可能用得上也可能用不上的指令。
  - 比如，处理器会预测哪个控制流更有可能被运行，再提取相应的指令代码和数据放到缓存中等待执行。若预测错误，不正确结果会被丢弃，处理器会恢复到预测执行前的状态，再重新跳转到正确执行的分支或指令中运行。

从网络上下载一个数据A，耗时30秒。
- 如果下载A成功，则直接给出A答案；
- 如果下载A失败，则计算30秒的算数B。

预测执行下：在A下载的30秒内，CPU是空转的，因此可以将B算出。这样如果将来需要到B，就节省了30秒的时间；就算不需要，丢弃B即可

- **乱序执行：** out of order，避免因为获取下一条程序指令所引起的处理器等待，无需等待直接处理下一条可以立即执行的指令。

```
1. b = a * 5
2. v = * b
3. c = a + 3
```

由于1与3可并发运行，而步骤2中b无法随即获得，因此可以先计算步骤1和3，再运行2

6.2.2 芯片的边信道攻击

- **乱序执行和预测执行漏洞本质原因：**乱序执行和预测执行对处理器缓存的操作会被保留。**被预测执行或乱序执行的指令及相关数据会被先加载到缓存中**，但在处理器恢复状态时并不会恢复处理器缓存的内容或刷新缓存的命中/失效状态，攻击者可以利用缓存边信道进行攻击。

- **乱序执行和预测执行经典漏洞**——Meltdown和Spectre
  - **Meltdown漏洞（熔炉）：**处理器中的关于预测执行机制的硬件安全漏洞，使得低权限的进程无论是否获取特权，均可以获取受高权限保护的内存数据
  - **Spectre漏洞（幽灵）：**存在于预测执行中的硬件安全漏洞，利用是时间旁路攻击，允许恶意进程获得其他程序在映射内存中的数据

6.2.2 芯片的边信道攻击

- **乱序执行和预测执行漏——Meltdown漏洞**

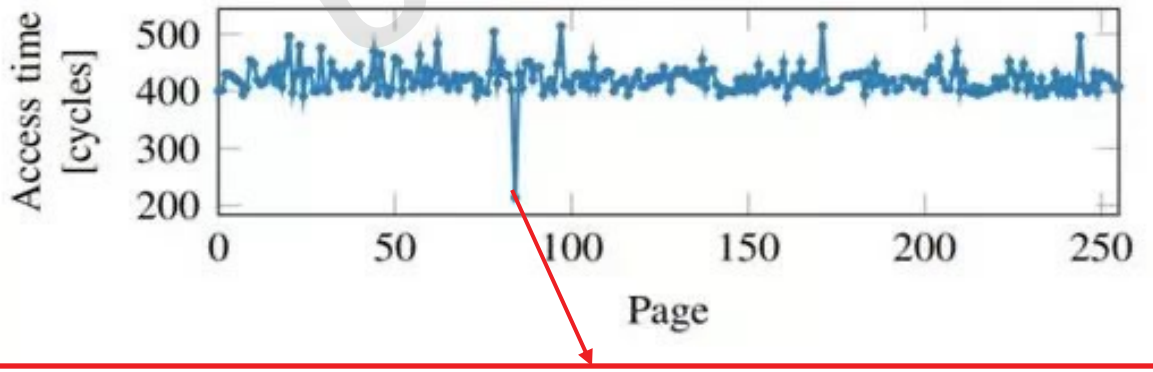
  核心代码分析：

```
1 mov rax byte[x] //将内核地址x处的内容存到寄存器rax非法操作。因为x内核地址，
                    读取byte[x]为非法操作，rax将被清零
2 shl rax 0xC // 内存页表对齐，Cache中存储页表，单位为4K字节
3 mov rbx Array[rax] // rax作为用户空间数组Array的index，对应的数据被写入rbx
```

  - 目标：读取内核地址x中机密数据（x为内核地址，受保护）
  - 理论上，在执行第2条指令前，由于非法访问，rax值应该已被清零
  - 然而，由于乱序执行，第2条和第3条指令在异常处理生效之前都会被**部分执行**，直到异常处理时 rax 和 rbx 才被清零
  - rax虽然无法被直接读取，但是已经存在，在CPU中可以用来计算

Meltdown漏洞（续）

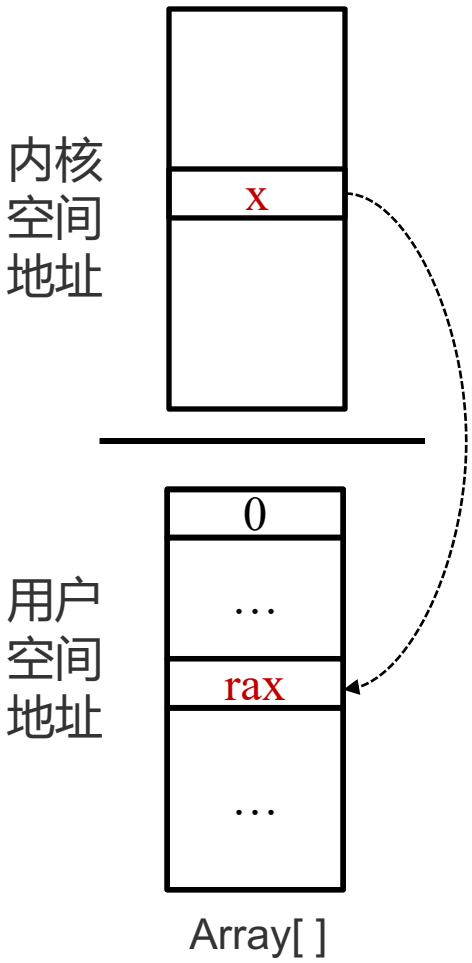
- 由于Array[]是在用户地址空间内的，可以自由操作；
- 如何获取rax：
  - 首先，要确保整个 Array[] 这个数组没有被Cache过；
  - 然后，执行上述攻击代码，这时候 rax代表的index就被Cache；
  - 遍历整个Array[]数组，测量内存访问时间，访问时间最短的page（rax代表的index）由于之前被cache过，因此访问时间最短，据此可以推测出rax值。



84 * page_size 处访问时间最短，因此确定原 rax 为84

Meltdown:https://meltdownattack.com/meltdown.pdf

Meltdown more explanation



mov rax byte[x] // rax值被保存，在异常处理之前，任然在CPU中可以用来后续计算；

mov rbx Array[rax] // 以rax值为index，将Array[rax]的结果数据写入rbx。导致Array[rax]结果被cache到缓存中

遍历数组Array[]，由于Array[rax]被cache过，因此rax对应的index读取时间最短，即可确定rax值

6.2.2 芯片的边信道攻击

- **乱序执行和预测执行漏洞—Spectre漏洞**

  目标：获取array1数组边界范围外的数据

```
1 if(x < size_of(array1)) {
2     y = array2[array1[x] * 256];
3 }
```

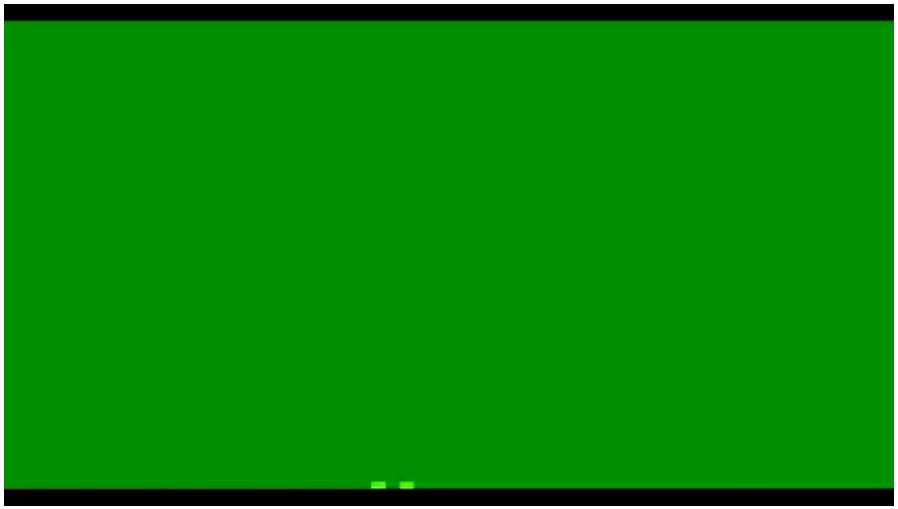
  - 理论上，如果x越界，即x>size_of(array1)，第2行代码不会被执行。但是，由于预测执行存在，即使x超出了array1范围，它还是会**“预测执行”**；
  - 一旦这个预执行被执行，攻击者就可以通过控制x的大小，让array2[]数组中index为array1[x]的数据被cache；
  - 利用缓存边信道，遍历整个array2[]数组，通过监测数据读取时间，就可以获得array1[]数组边界范围外的数据

Spectre:https://spectreattack.com/spectre.pdf

Meltdown & Spectre demo video

Demo 1:
Spying on user password

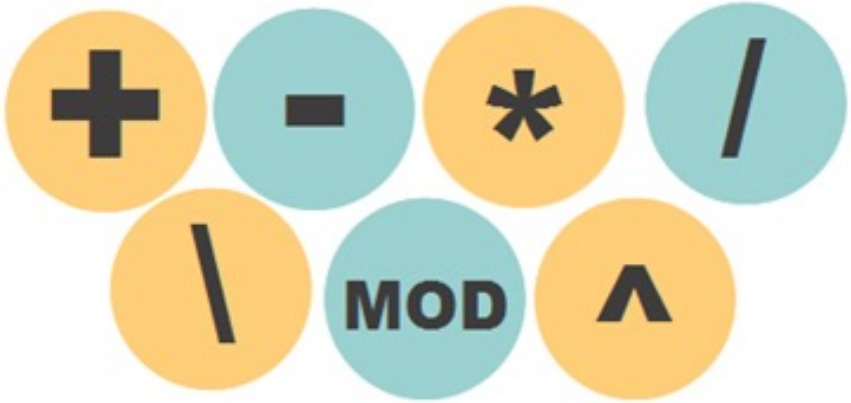
Demo 2:
Reconstructing a photo



https://meltdownattack.com/

6.2.2 芯片的边信道攻击

- **时间边信道攻击——定义**
  - 通过程序运算用时来推断所使用的运算操作
  - 通过对比运算的时间推测隐私信息



6.2.2 芯片的边信道攻击

- **示例：窃取芯片中神经网络激活函数**
  - 原理：不同激活函数的计算运行时间存在差异

激活函数	计算时间（ns）
$f_{\mathrm{ReLU}}(x) = \max(0, x)$	5975
$f_{\mathrm{sigmoid}}(x) = \frac{1}{1+e^{-x}}$	184864
$f_{\mathrm{softmax}}(x_i) = \frac{e^{x_i}}{\sum_{k=1}^{K} e^{x_k}}, k = 1,2,\ldots,K$	813712

Lejla Batina, etc., CSI NN: Reverse Engineering of Neural Network Architectures Through Electromagnetic Side Channel, in USENIX Security 2019

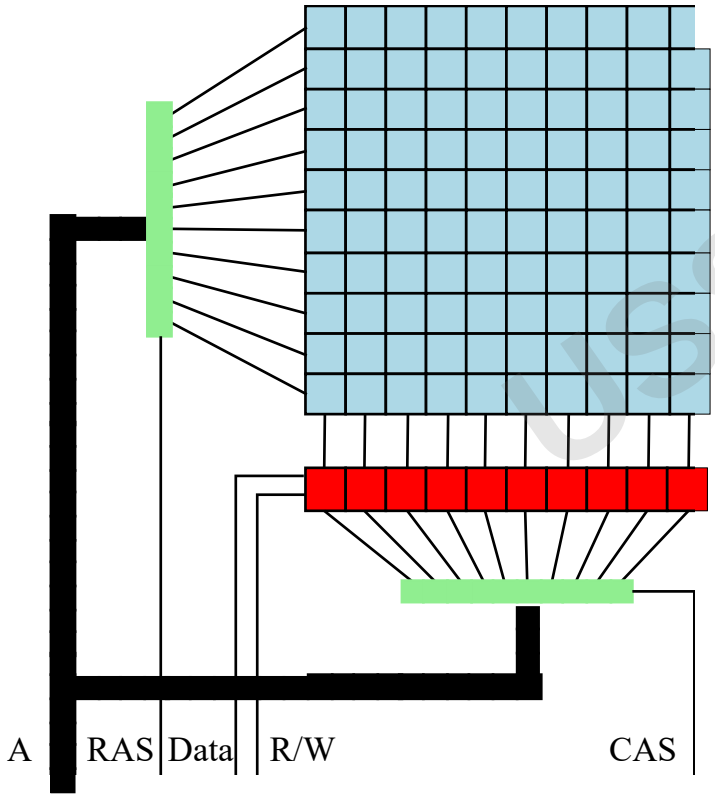
6.2.3 存储器读取攻击——行锤攻击

- **定义**：动态随机存取存储器（DRAM）中的意外和不良副作用，即存储器单元的电荷泄露和单元之间的电气相互作用，造成泄漏或改变附近存储器行的内容的一种攻击

- **本质原因**：现代DRAM中的存储器单元高密度集成，并且可以通过特殊的存储器访问模式触发，这些模式可以快速激活相同的存储器行数次

6.2.3 存储行锤攻击

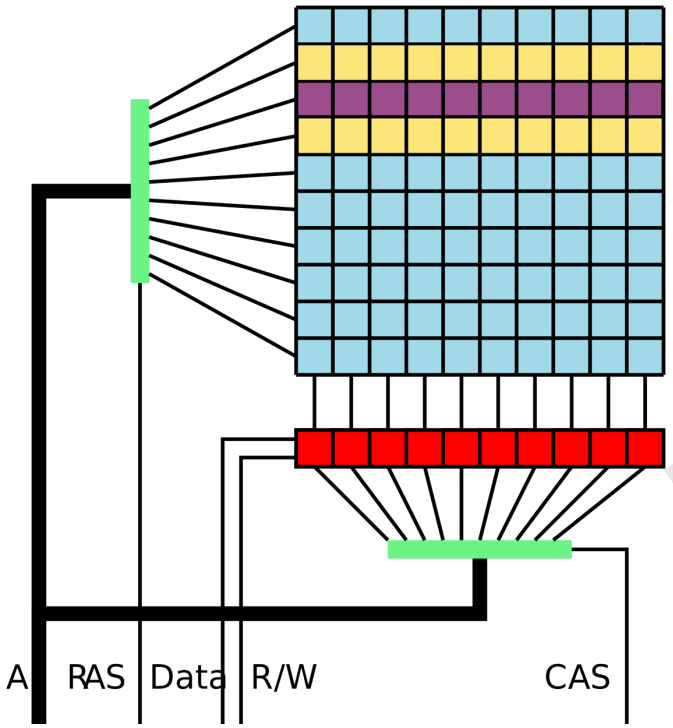
- **行锤攻击**



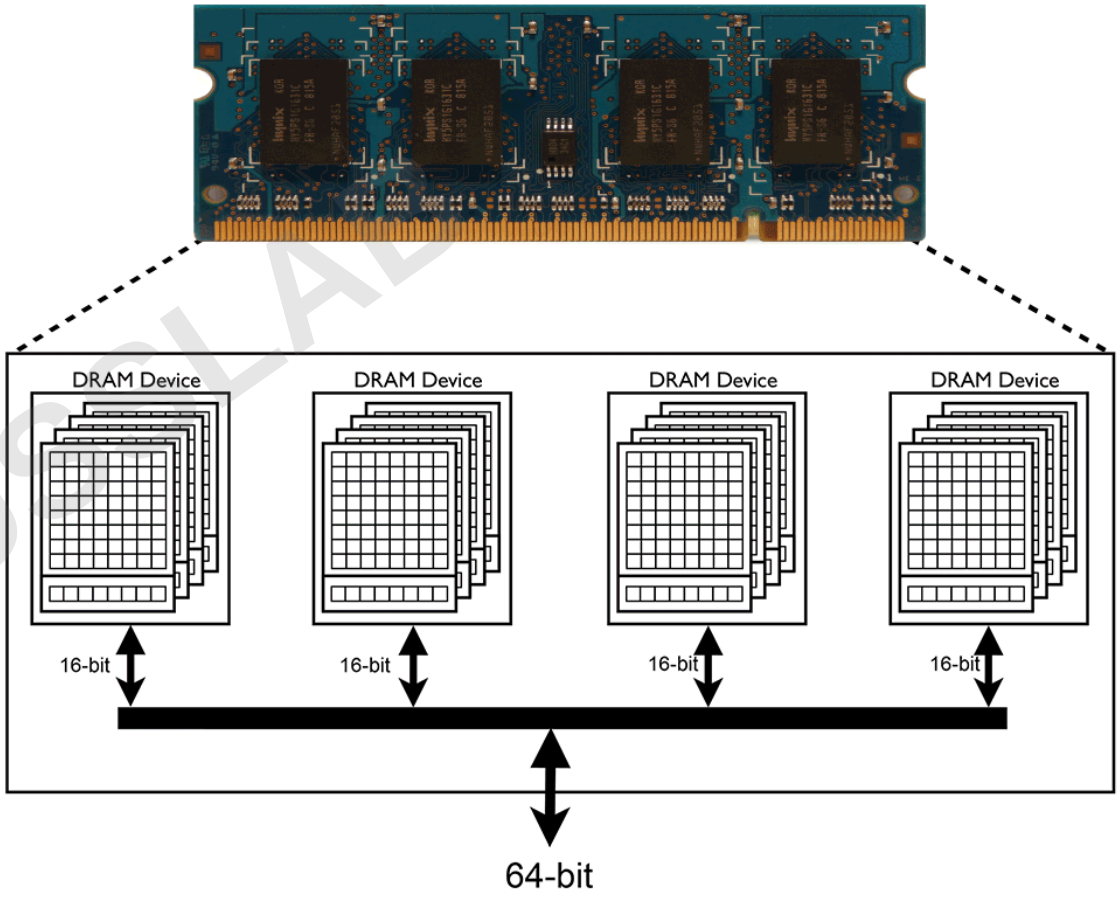
DRAM结构图，包括存储单元（蓝色方块），地址解码器（绿色矩形）和读出放大器（红色方块）

- 存储单元的地址分为行地址和列地址，分别由行和列地址解码器处理。选中某一行后，该行中所有单元都被传输到读出放大器，再使用列地址选中某一位；
- 每个存储单元由一个电容器和一个晶体管实现，由于电容的自然放电，所有存储单元都需要定期重写；
- 频繁的行激活会导致相邻行的电压波动，提高其放电率，若受影响的存储器单元在它们失去太多电荷之前没有被刷新，则会发生**干扰错误**

行锤攻击演示



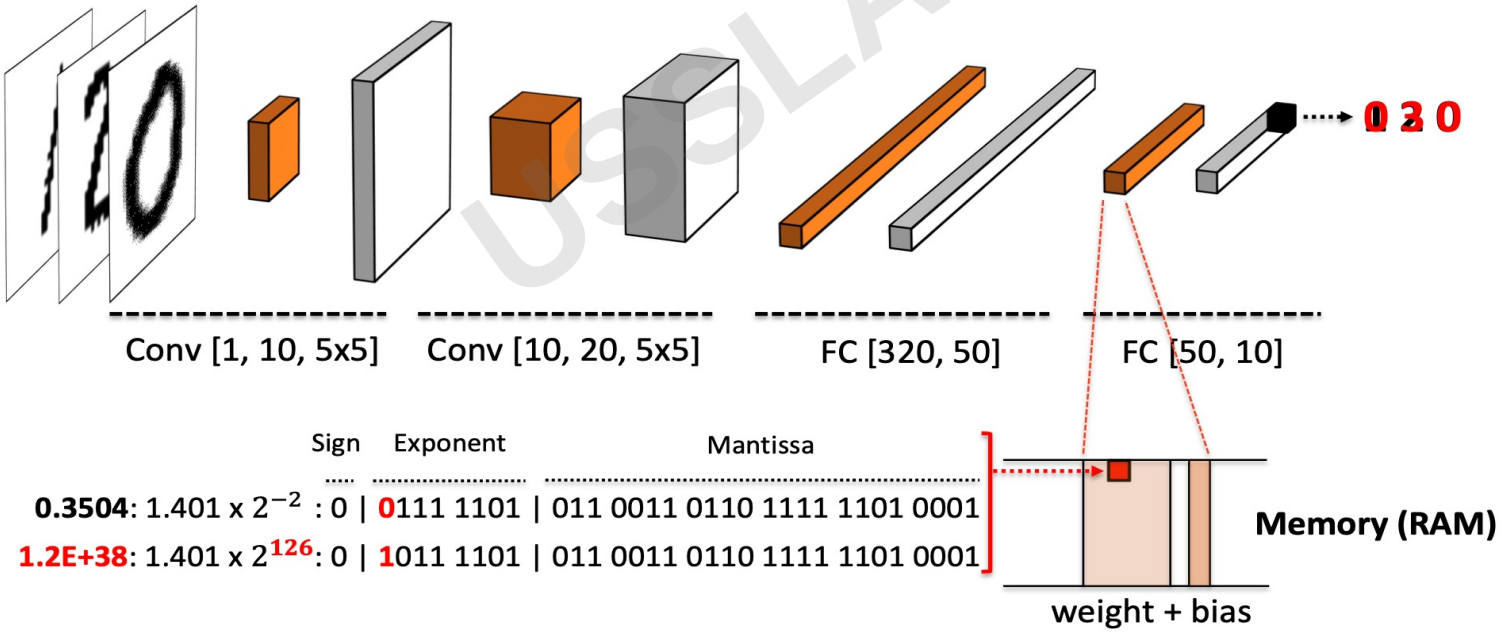
对黄色行的频繁激活导致粉色行的数据位的值发生改变



Row-hammer后续

- 利用Row-hammer攻击，造成内存中神经网络的参数发生位翻转，导致神经网络分类精度下降，甚至不可用

  - Accuracy: **57.52% (41.01% drop)**



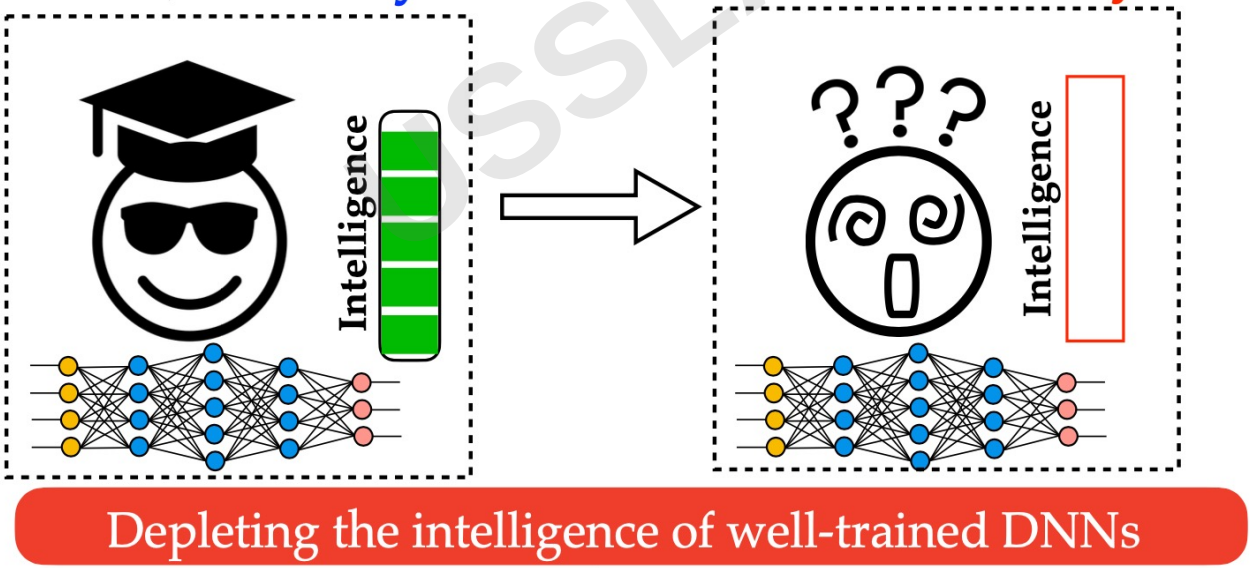
Sanghyun Hong etc., Terminal Brain Damage: Exposing the Graceless Degradation in Deep Neural Networks Under Hardware Fault Attacks, in USENIX Security 2019

Row-hammer后续

- 基于上个工作，如何更好定位脆弱bit并进行攻击？[1]
- 仅翻转13/93 million bits，分类性能从68%→0.1%。[2]

Example: ResNet-20 for CIFAR-10, **10** output classes

Before attack, **Accuracy: 90.2%** After attack, **Accuracy: ~10% (1/10)**



[1] Adnan Siraj Rakin, etc., Bit-Flip Attack: Crushing Neural Network with Progressive Bit Search, in ICCV 2019
[2] Fan Yao, etc., DeepHammer: Depleting the Intelligence of Deep Neural Networks through Targeted Chain of Bit Flips, in USENIX Security 2020

6.3 芯片安全与防护

6.3.1 芯片安全与防护——PUF

- 物理不可克隆函数(physical unclonable function, PUF)：指芯片制造过程中必然引入的工艺参数偏差，从而导致**激励信号与响应信号之间具有的唯一对应的函数关系**
- **本质原理**：同种电路的时延有微小的差别，两路开关信号到达的时间不同导致输出结果不同。任何探测或观测PUF操作的尝试都将改变底层电路特征，可以防范侵入式物理攻击。
- PUF具有以下属性：
  - 长期不变：不同的时间、温度和工作电压条件中保持不变
  - 不可读取：任何探测或观测PUF操作的尝试都将改变底层电路特征

Q：PUF和硬件指纹有何区别？

6.3.1 芯片安全与防护——PUF

- **PUF的应用**

  - **加密密钥存储**：将密钥存储在芯片独特的内在物理特征中，将PUF输出的不可直接读取的唯一值作为私钥

  - **设备认证**：PUF的“激励—响应”机制，可以用于低成本的设备认证

  - **FPGA的IP保护**：将硬件的不可克隆PUF签名与FPGA设计/IP中的时序电路的现有状态机（FSM）相结合

6.3.1 芯片安全与防护——TPM/TEE

硬件协助的芯片安全，包括：

- 可信平台模块（Trusted Platform Module, TPM）
  - 一种专门用来加密的硬件
  - 独立的安全芯片
  - 物理上与处理系统的其余部分隔离
  - 具有固定的功能，无法增加或更改

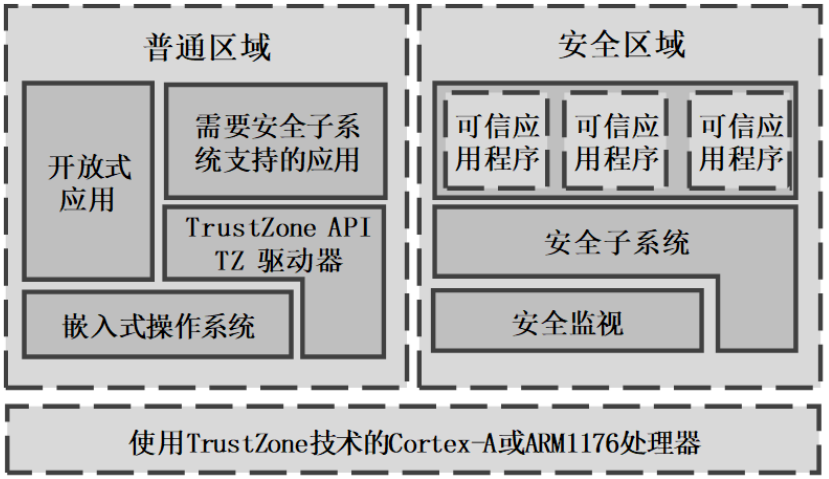
- **可信执行环境（Trusted Execution Environment, TEE）**
  - 软件和硬件的组合
  - 芯片组上的一个区域
  - 物理上与芯片其他部分不隔离
  - 作为隔离的执行环境，运行的代码由开发人员自由决定

6.3.1 芯片安全与防护——TEE

- **典型TEE——ARM TrustZone**
- 由ARM公司开发的一种面向处理器的安全技术，通过构建可编程环境，使得代码和数据的机密性和完整性都得到保护，可以不受主操作系统中运行的用户安装应用程序影响。
- TrustZone对SoC的硬件和软件资源进行分区，分成**安全区域**和**普通区域**，并通过**硬件方式支持访问控制和权限**，以处理安全/非安全应用程序以及它们之间的交互和通信。
- 访问单向受限、双区硬件隔离。

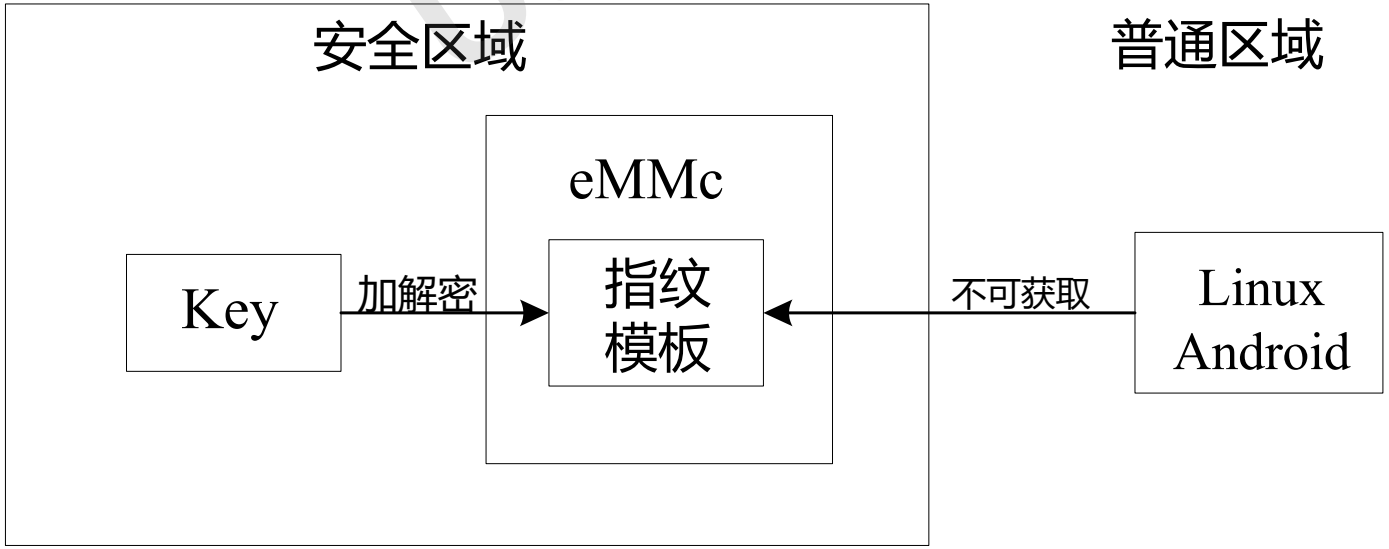
普通区域：Android, iOS及各类应用程序

安全区域：各类隐私、机密数据，如指纹等



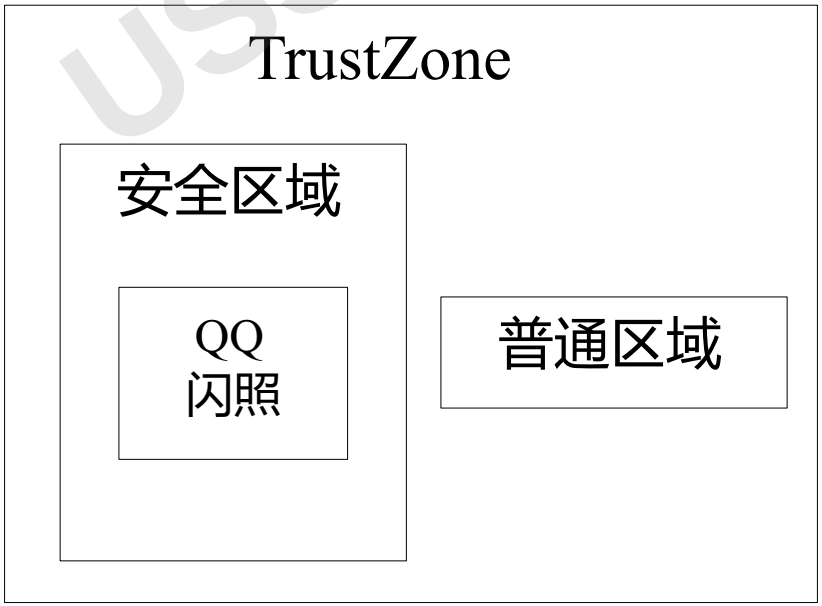
6.3.1 芯片安全与防护

- ARM TrustZone应用——保存手机指纹模板
  - TrustZone规定SoC制造商需要实现一套只有安全空间才能访问的密钥存储技术（比如，可通过集成TPM芯片保存密钥）。
  - 在安全区域中，使用密钥对指纹模板进行加密，保存在手机的eMMC(Embedded Multi Media Card)上，类似于SD卡。
  - 在普通区域中，无法获得加密密钥，无法解密指纹模板。



6.3.1 芯片安全与防护

- ARM TrustZone的应用——QQ闪照技术
  - QQ闪照：一张照片只能点开一次（阅后即焚），不可保存或截图
  - 原理：QQ闪照储存在TrustZone划分的SoC安全区中，这个区域中运行的代码只有特定软件自己可读，即只有QQ和屏幕输出才能读取图片，其他程序都不可以。

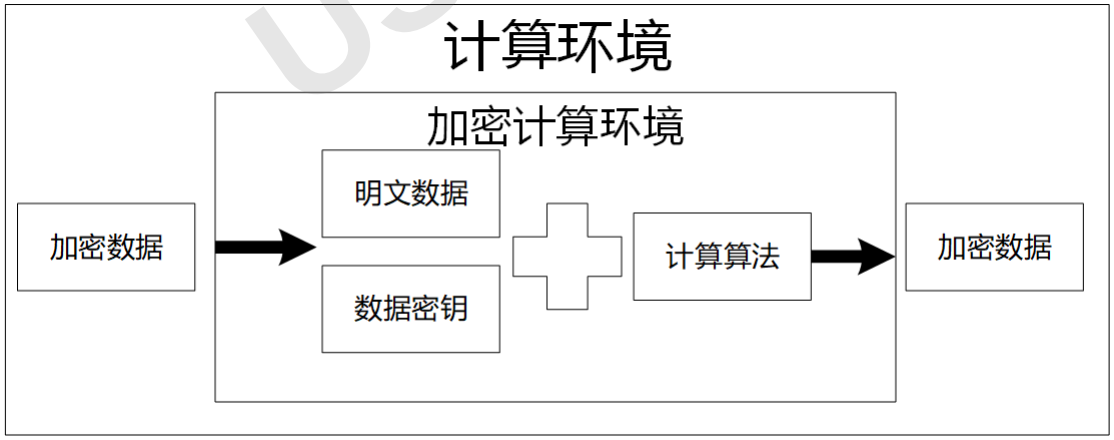


6.3.1 芯片安全与防护

- **典型TEE——Intel SGX**
  - SGX（Intel Software Guard Extension）是英特尔指令集架构的一个扩展。SGX会提供飞地（Enclave），即内存中一个加密的可信执行区域，由CPU保护个人的数据和隐私不被恶意代码窃取
  - **内存加密**：当数据从飞地的内部内存传输到外部内存时，SGX通过CPU内的加密引擎MEE (Memory Encryption Engine)对数据进行加密。EPC中加密的内容只有进入CPU后才会被解密成明文
  - 因此，在SGX中，用户可以不信任操作系统、VMM、BIOS，只需要信任CPU便能确保隐私数据不会泄漏

6.3.1 芯片安全与防护

- **Intel SGX的应用**
  - 案例：密码管理软件场景。用户在密码管理软件如1Password中输密码后，密码数据会在该软件的运行时的内存中出现。为防止拥有操作系统权限的恶意程序在内存中窥探密码数据，Intel SGX技术可以将密码数据和加解密过程放在安全区，即使操作系统管理员权限也无权访问篡改



  - 使用场景：云端医疗、语音识别等

6.3.1 芯片安全与防护

- **其他通用芯片防护技术**
  - **限制程序计数器**：攻击者可以利用程序计数器来增加对内存数据的访问。
  - **随机时钟信号**：许多非攻击性技术是要攻击者预见某条指令执行的准确时间。
  - **多层的电路设计**：增加电路成熟，从而增加攻击的难度。
  - **自毁技术**：给芯片穿上保护衣，如攻击者用精密机械探针插入芯片内企图探测里面的密码，会引起短路而烧毁芯片。
  - **抗电磁探测密码技术**：对电磁辐射进行屏蔽、衰减或干扰。
  - **锁存电路**：出现异常情况时，清除芯片中的敏感数据。
  - ……

6.4 本章总结

本章总结

- 了解芯片的制造流程
- 了解芯片安全的引入因素
- 掌握芯片木马的原理和利用思路
- 理解并掌握芯片边信道攻击和行锤攻击

课后作业

1、简答题：请简述行锤攻击的原理。

2、简答题：请列出至少三种针对芯片的攻击。

3、判断题：Meltdown与Spectre从本质上来看属于利用处理器的乱序执行或预测执行漏洞进行的缓存边信道攻击。

4、设计题：如何利用手机的磁力计检测其他应用程序对于麦克风传感器的滥用，比如百度输入法滥用麦克风。