

OpenRigil: 开源 RISC-V 密码学硬件密钥

郑鉉壬¹ 叶泽文² 刘玖阳³ 党凡¹ 高鸣宇¹

¹ 清华大学

² 浙江大学

³ 华中科技大学

2022-08-25

- 各位看官，看到这个标题，你可能有两个疑问：
 - 1 何为「密码学」「硬件密钥」
 - 2 这与「开源」「RISC-V」有啥关系

目录

- 密码学硬件密钥
- 开源与 RISC-V
- 现有开源项目
- 我们的工作
 - 实现 RISC-V 标量密码学扩展
 - 蒙哥马利模乘加速器
 - USB 1.1 外设
 - 完整系统搭建
- FPGA 原型与实验
- 缺陷与讨论

1

密码学硬件密钥

第一印象

- 大家可能对这种设备比较陌生
- 第一印象是 U 盘
- 但它并不能存一般的文件
- 那为什么会需要这种设备

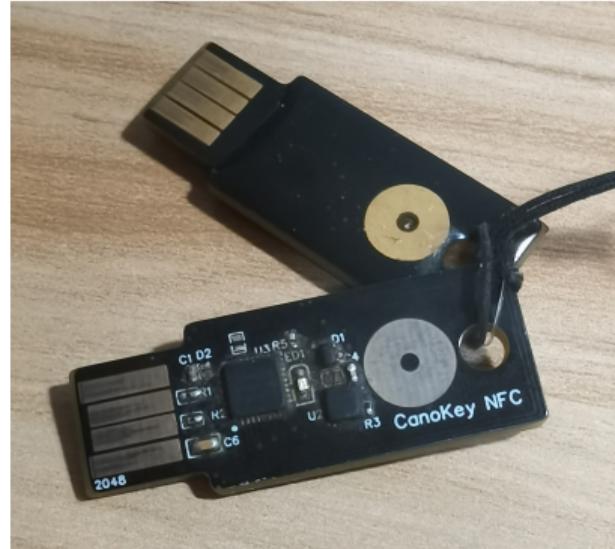


图: 密码学硬件密钥样例

从简单场景出发：密码

- 我们来说说帐号密码系统
- 密码是个麻烦的东西

从简单场景出发：密码

- 我们来说说帐号密码系统
- 密码是个麻烦的东西
- 想必大家对怎么设置自己各类帐号的密码都各有心得

从简单场景出发：密码

- 我们来说说帐号密码系统
- 密码是个麻烦的东西
- 想必大家对怎么设置自己各类帐号的密码都各有心得
 - 弱密码：容易爆破
 - 强密码：容易忘记
 - 忘记密码：找回密码很麻烦

从简单场景出发：密码

- 我们来说说帐号密码系统
- 密码是个麻烦的东西
- 想必大家对怎么设置自己各类帐号的密码都各有心得
 - 弱密码：容易爆破
 - 强密码：容易忘记
 - 忘记密码：找回密码很麻烦
 - 各帐号同一个密码：容易泄漏

从简单场景出发：密码

- 我们来说说帐号密码系统
- 密码是个麻烦的东西
- 想必大家对怎么设置自己各类帐号的密码都各有心得
 - 弱密码：容易爆破
 - 强密码：容易忘记
 - 忘记密码：找回密码很麻烦
 - 各帐号同一个密码：容易泄漏
 - 将密码记在小本本上：不是良好习惯
 - 密码管理器：各个设备之间的同步，管理器、服务商是否可信

解决密码的麻烦：多因子验证

- 多因子验证 (MFA): 密码 + 其他手段
(因子)



图: 验证码样例

解决密码的麻烦：多因子验证

- 多因子验证 (MFA)：密码 + 其他手段 (因子)
- 多因子验证已经被实践了很久
 - 短信验证码
 - 扫码登录 (其他帐号验证)
 - 生物因素 (人脸识别、指纹)
 - 软件验证码 (Microsoft Authenticator)



图: 验证码样例

解决密码的麻烦：多因子验证

- 多因子验证（MFA）：密码 + 其他手段（因子）
- 多因子验证已经被实践了很久
 - 短信验证码
 - 扫码登录（其他帐号验证）
 - 生物因素（人脸识别、指纹）
 - 软件验证码（Microsoft Authenticator）
- 还有没有别的因子呢？公钥密码学



图: 验证码样例

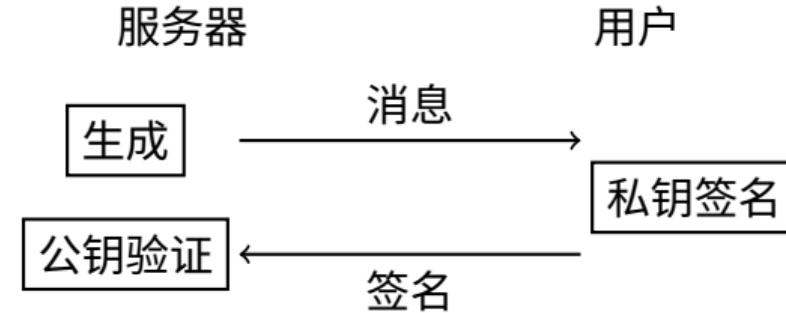
另一个场景：电子签名

- 签名自古以来的难题：伪造
 - 手写签字
 - 盖章
- 公钥密码学签名
 - 密码学认为难以伪造



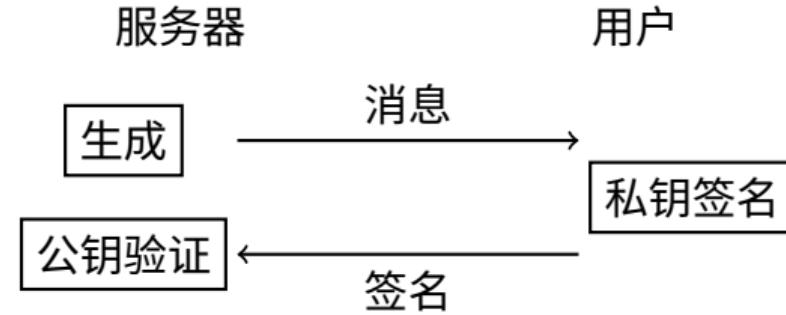
^a图自《攻壳机动队》，图文无关

公钥密码学



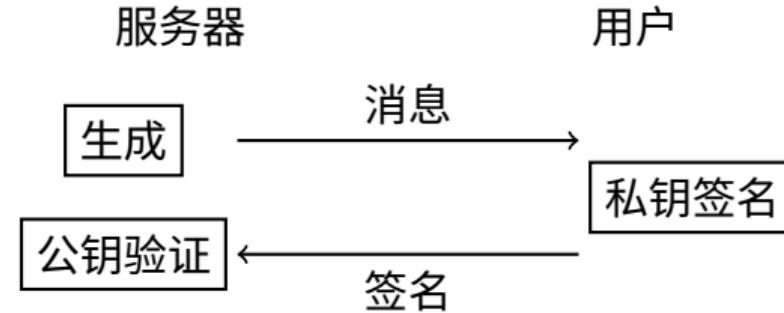
- 注册：用户生成私钥与公钥，并讲公钥提交给服务器

公钥密码学



- 注册：用户生成私钥与公钥，并讲公钥提交给服务器
- 登录：服务器生成消息，让用户签名，如果该签名能通过公钥检验，则可以登录

公钥密码学



- 注册：用户生成私钥与公钥，并将公钥提交给服务器
- 登录：服务器生成消息，让用户签名，如果该签名能通过公钥检验，则可以登录
- 密码学认为：难以通过公钥计算私钥，难以伪造签名
- 常见公钥密码学实现：RSA 与 ECC（椭圆曲线密码学）

公钥密码学很好，但还不够

- 用户私钥怎么存放，怎么管理

公钥密码学很好，但还不够

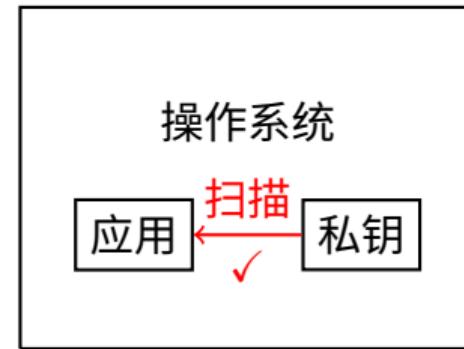
- 用户私钥怎么存放，怎么管理
- 人脑记忆、人脑计算
 - 人脑算 RSA4096
 - 请您上最强大脑！



图: 最强大脑

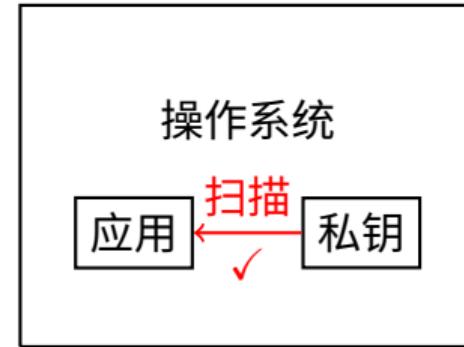
公钥密码学很好，但还不够

- 用户私钥怎么存放，怎么管理
- 人脑记忆、人脑计算
 - 人脑算 RSA4096
 - 请您上最强大脑！
- 私钥存在电脑上，应用旁边
 - 那应用会不会偷偷读取你的私钥
 - 闭源软件最终会演变成恶意软件
 - 假如软件偷偷全盘扫描……



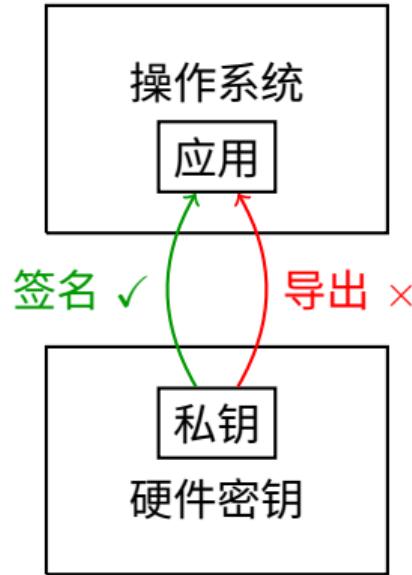
公钥密码学很好，但还不够

- 用户私钥怎么存放，怎么管理
- 人脑记忆、人脑计算
 - 人脑算 RSA4096
 - 请您上最强大脑！
- 私钥存在电脑上，应用旁边
 - 那应用会不会偷偷读取你的私钥
 - 闭源软件最终会演变成恶意软件
 - 假如软件偷偷全盘扫描……
- 如何解决



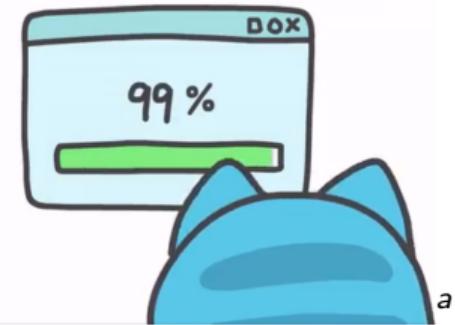
硬件密钥

- 将私钥放在「硬件密钥」上
- 接口保证不导出私钥，只允许签名
- 签名的时候需要用户交互
- 保护了私钥



「密码学」硬件密钥

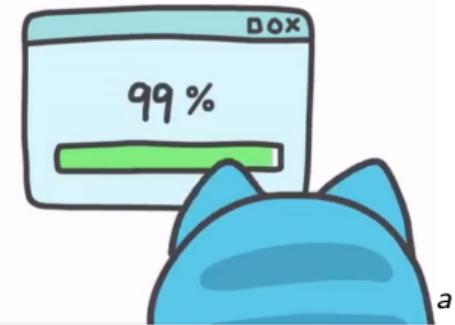
- 密码学运算需要的计算量大
- 例如不使用硬件加速在嵌入式芯片或 FPGA 上用 RSA4096 签名，需要 20 秒



^aCapoo

「密码学」硬件密钥

- 密码学运算需要的计算量大
- 例如不使用硬件加速在嵌入式芯片或 FPGA 上用 RSA4096 签名，需要 20 秒
- 想象你正在赶 DDL，23:59 交作业
- 现在是 23:58，你正在登录系统，事态非常紧急
- 结果登录的时候在等签名……



^aCapoo

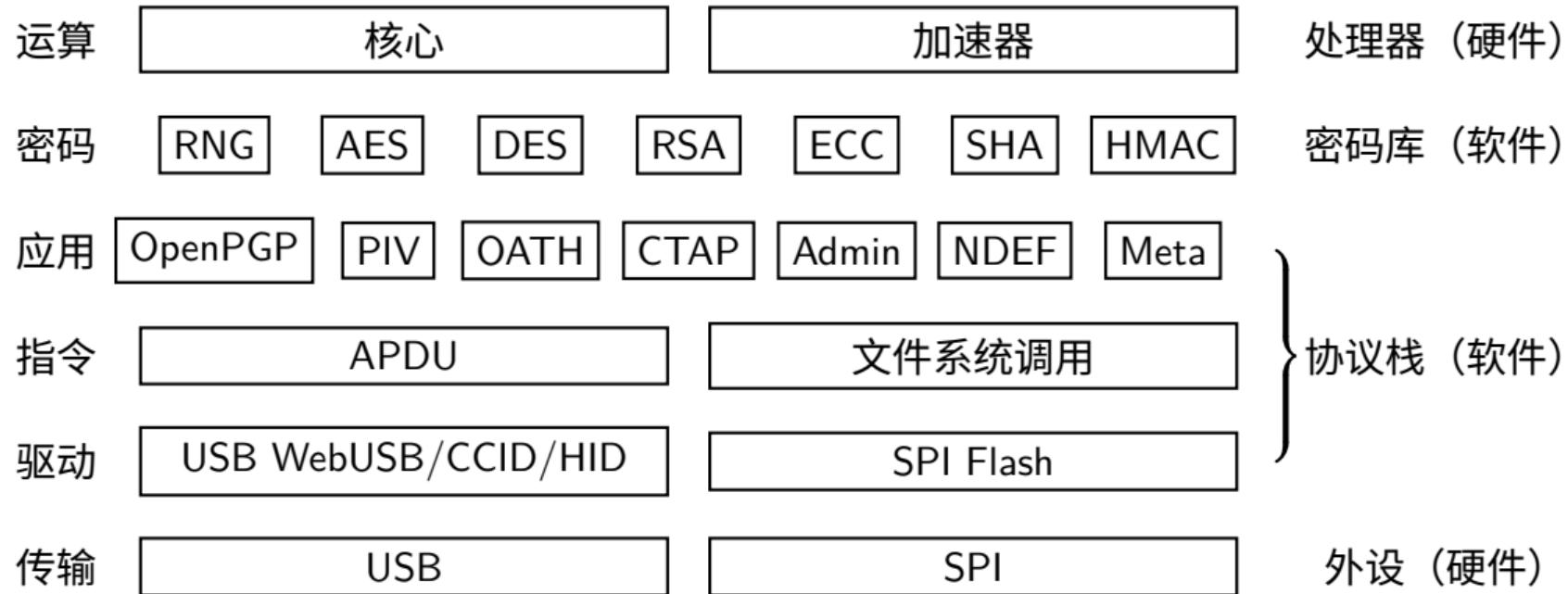
「密码学」硬件密钥

- 密码学运算需要的计算量大
- 例如不使用硬件加速在嵌入式芯片或 FPGA 上用 RSA4096 签名，需要 20 秒
- 想象你正在赶 DDL，23:59 交作业
- 现在是 23:58，你正在登录系统，事态非常紧急
- 结果登录的时候在等签名……
- 所以需要「密码学」硬件密钥
- 即搭载密码学加速器的硬件密钥

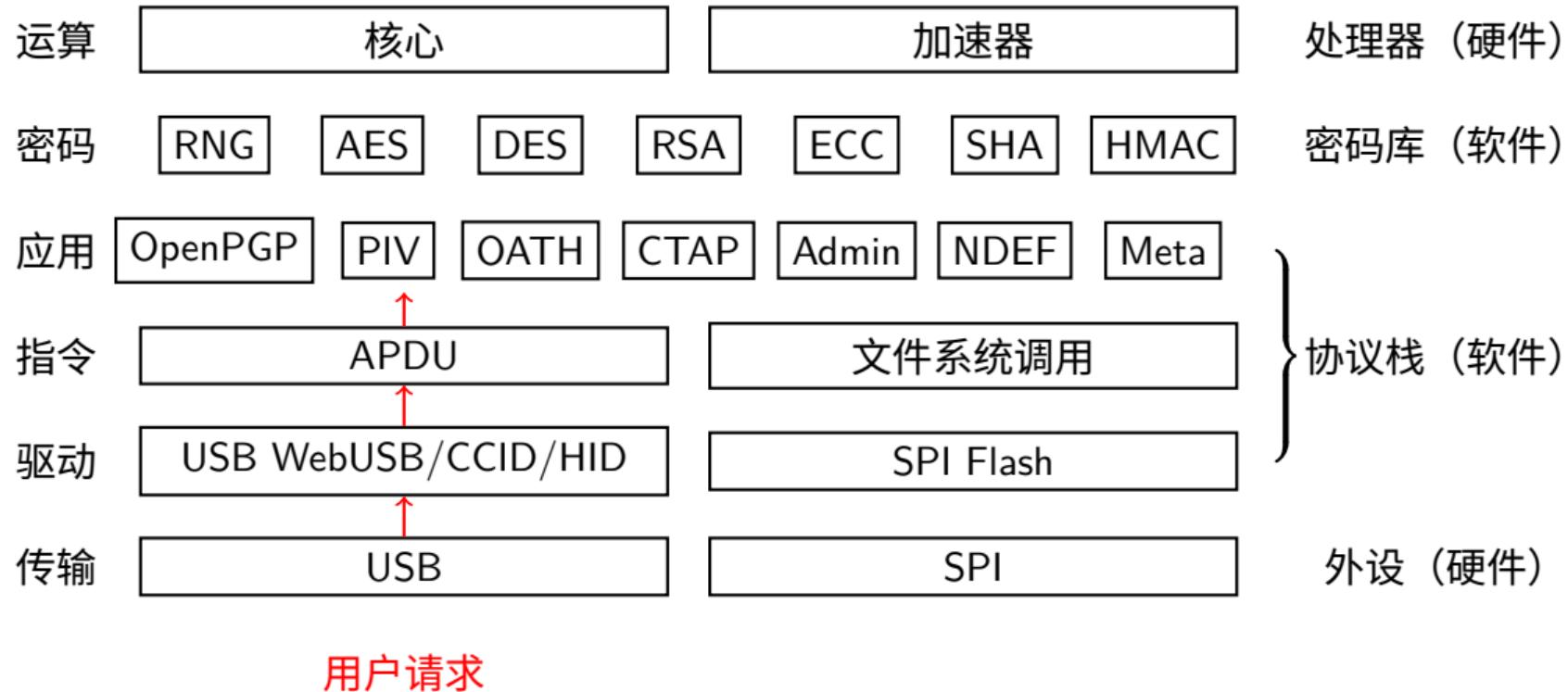


^aCapoo

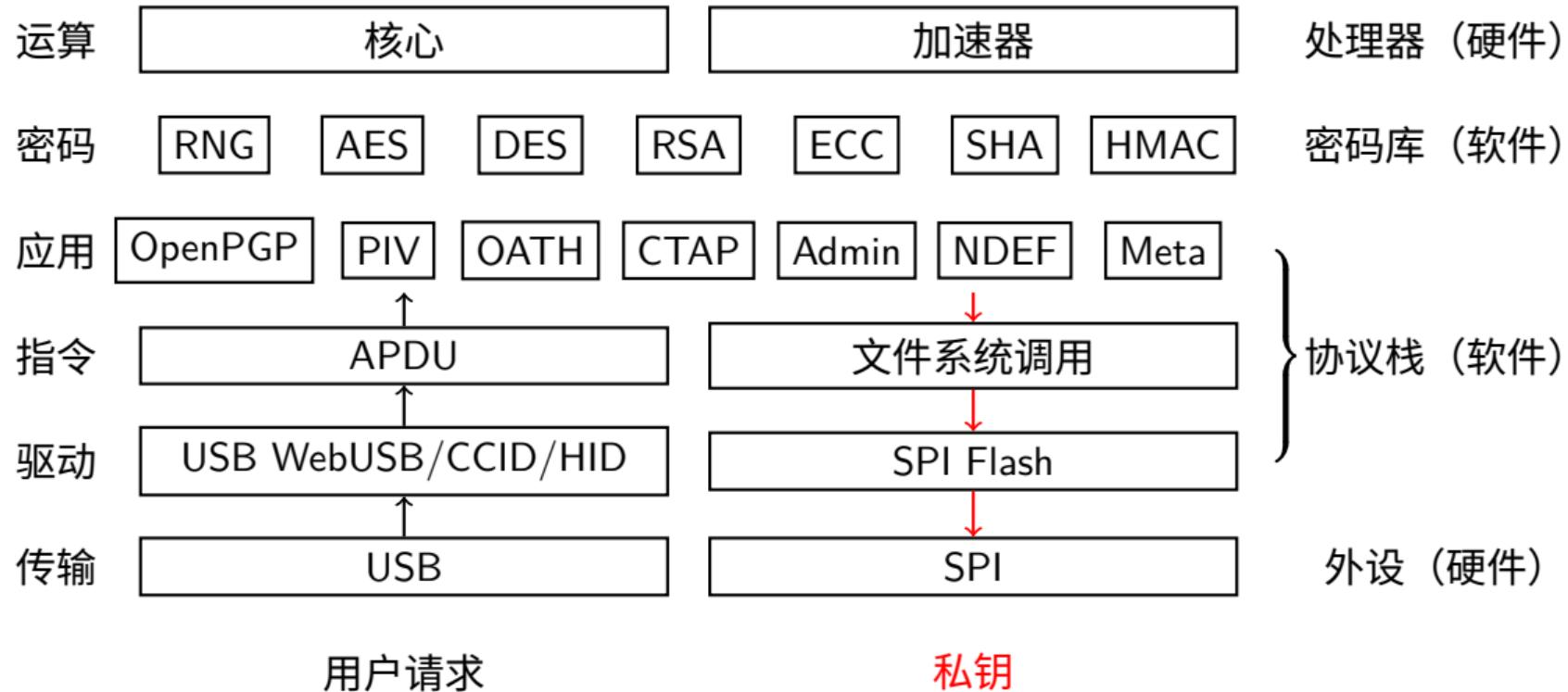
密码学硬件密钥架构



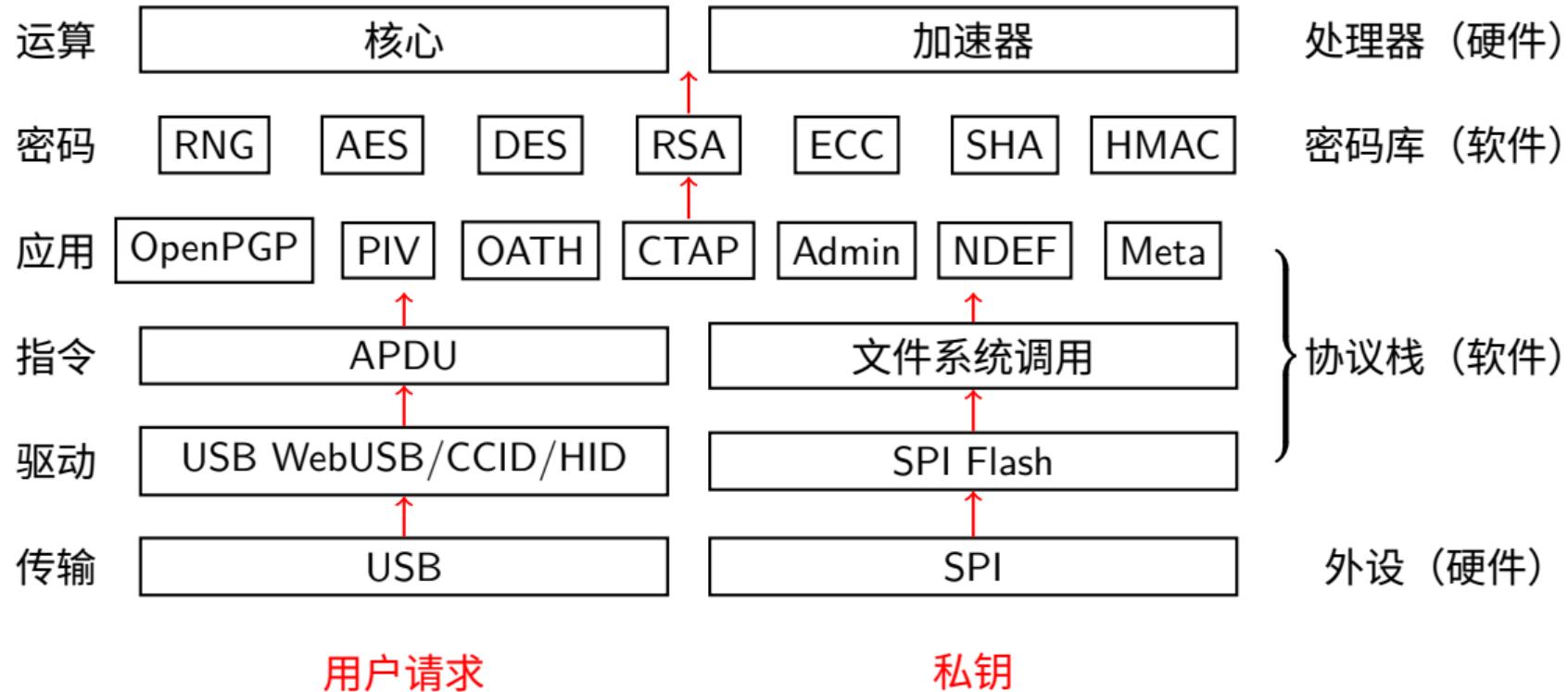
密码学硬件密钥架构



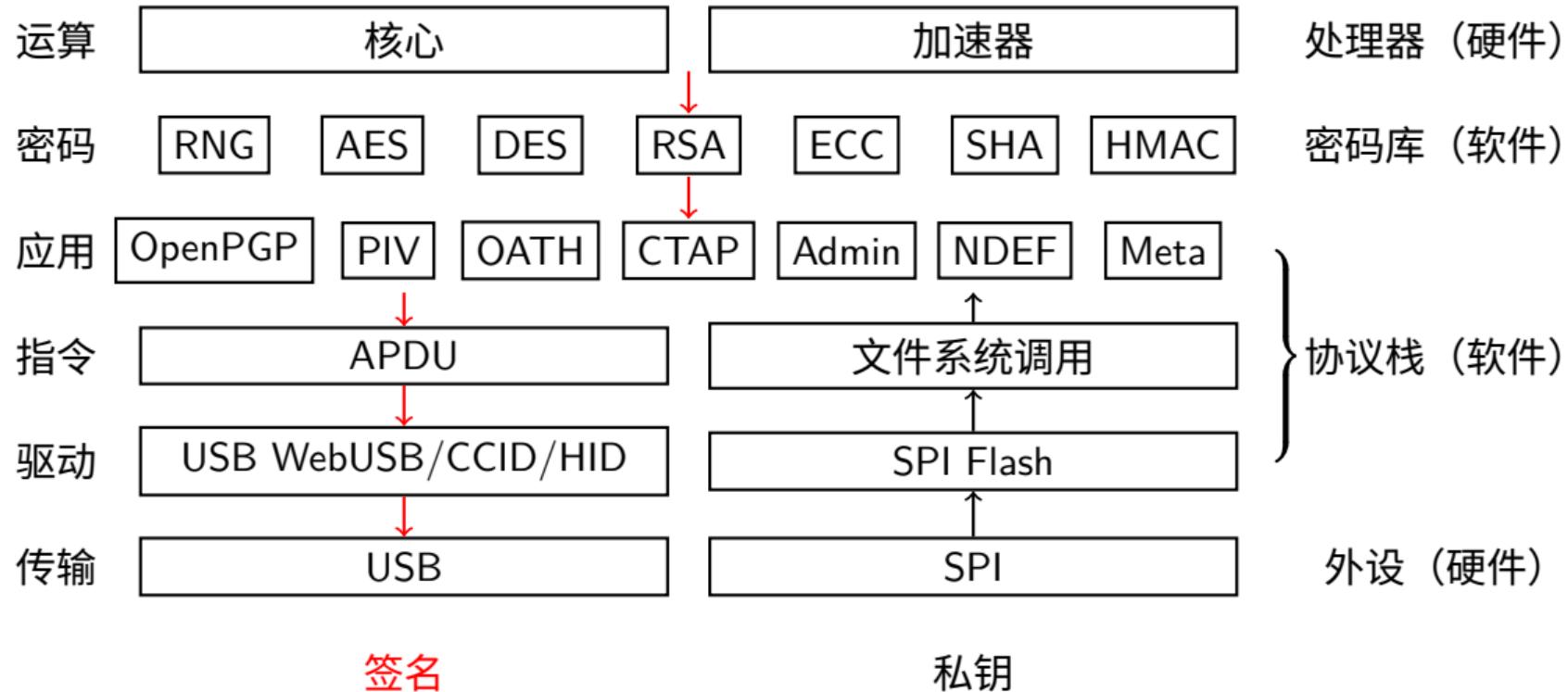
密码学硬件密钥架构



密码学硬件密钥架构



密码学硬件密钥架构



2

开源与 RISC-V

现有产品：不够开源

- 我来峰会，只办三件事：开源、开源、还是开源
- 之前说到，密码学硬件密钥保护了私钥
- 这是建立在密码学硬件密钥「可信」的基础上的
- 我认为「开源」是「可信」的最低要求
 - 可供用户与开发者审计，找出潜在的问题
 - 能较大程度上避免植入恶意后门

现有产品：不够开源

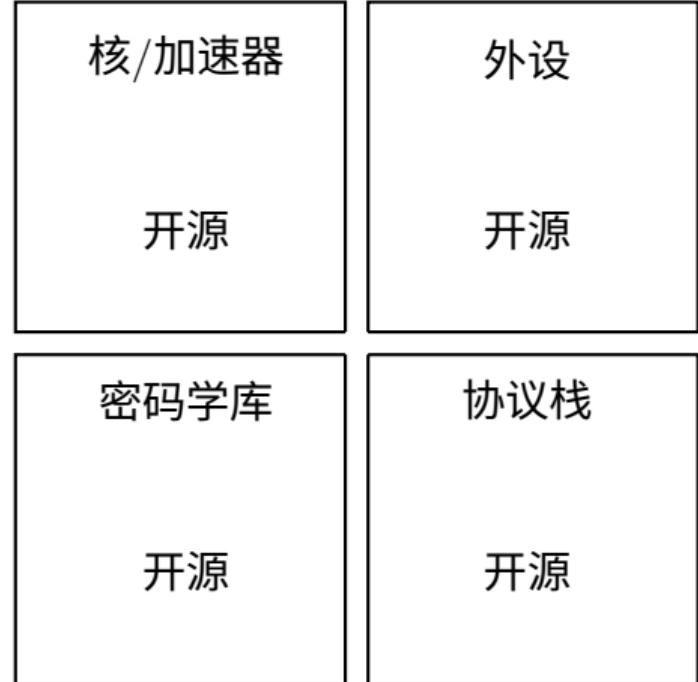
- 我来峰会，只办三件事：开源、开源、还是开源
- 之前说到，密码学硬件密钥保护了私钥
- 这是建立在密码学硬件密钥「可信」的基础上的
- 我认为「开源」是「可信」的最低要求
 - 可供用户与开发者审计，找出潜在的问题
 - 能较大程度上避免植入恶意后门
- 现有产品最多做到固件开源，硬件开源难以做到
 - YubiKey：核，加速器，固件均不开源
 - STM32 阵营（NitroKey/SoloKey）：固件开源但 ARM 核不开源
 - nRF52 阵营（OpenSK）：ARM 核、密码学加速器不开源

RISC-V 的天然优势

- RISC-V 峰会，当然还是要吹一波 RISC-V 的
- 不过说实话，这个项目能成，主要还是 RISC-V 有基础设施
- 我们能够自由使用 RISC-V ISA 和相应工具链
- 我们能够自由选择开源 RISC-V 核，例如 rocket-chip, cva6
- 相关的开源外设也较为方便使用

我们的目标

- 我们要做开源硬件密钥
- 我们追求：全部开源
 - RISC-V 核心开源
 - 加速器开源
 - 外设开源
 - 密码学库开源
 - 协议栈开源



我们的限制

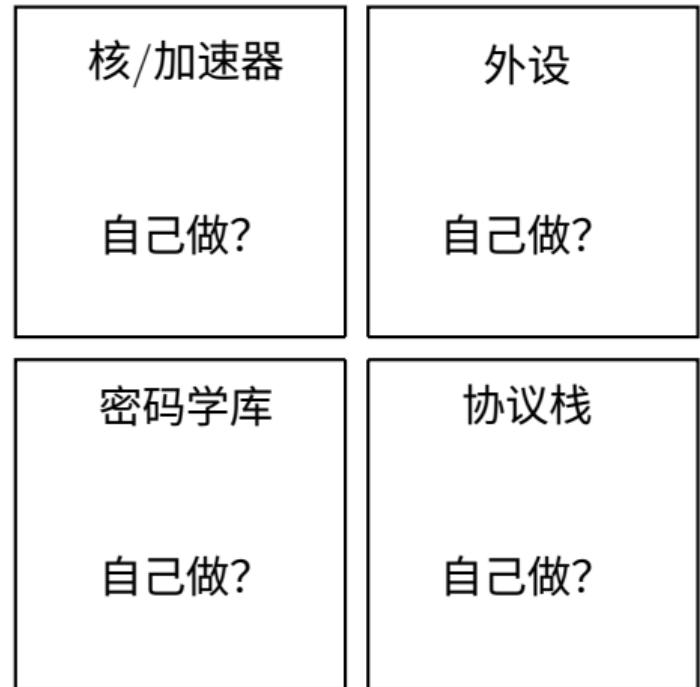
- 时间和人力有限：几个人业余做
 - 以较小的工作，搭建一个跑通的系统，展现这是能做的
- 不求尽善尽美
 - 功能不一定最全，性能不一定最优
 - (比不过比不过，毕竟别人是靠这个吃饭的)
- 在峰会上介绍给社区，社区可复现
 - 开源的好处：想加功能，想优化，都可以动手做

3

现有开源项目

不可能平地起高楼

- 自己写核，自己写总线，自己写外设，自己写软件，自己写密码学库，自己写协议栈，自己调试...
- 工期来不及
 - 不做个三五年怎么可能做完
 - 三年综合、五年模拟
- 时效性不行
- 一定要复用现有开源软件，快速出原型



还差那么……一点点

- 一些观众：这不是随便抓一些项目来就能跑通
- 我们来假想一下要选哪些
 - 核：rocket chip, cva6, ibex...

核/加速器	外设
RV32IMACZk MMM 加速器	USB SPI UART
密码学库	协议栈
RSA ECC AES SHA	FIDO2 OATH OpenPGP PIV

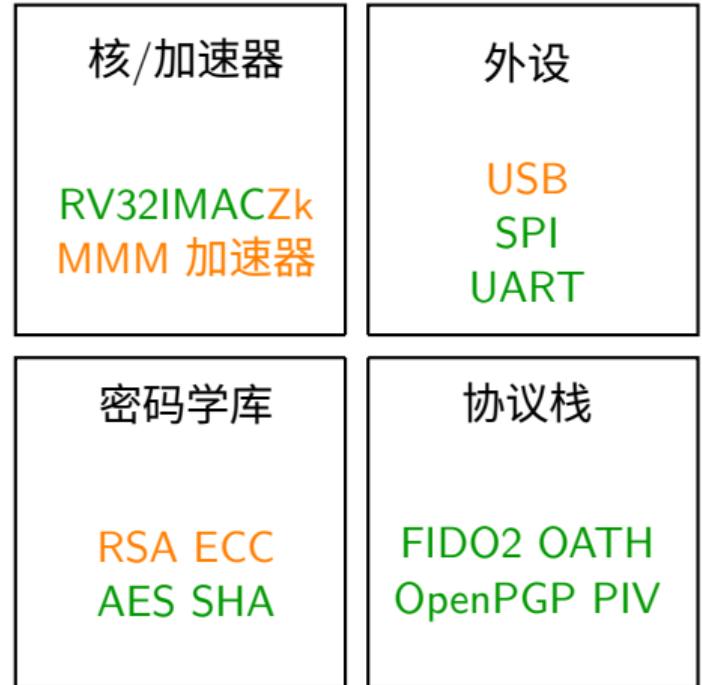
还差那么……一点点

- 一些观众：这不是随便抓一些项目来就能跑通
- 我们来假想一下要选哪些
 - 核：rocket chip, cva6, ibex...
 - 加速器：并不完全符合需求

核/加速器	外设
RV32IMACZk MMM 加速器	USB SPI UART
密码学库	协议栈
RSA ECC AES SHA	FIDO2 OATH OpenPGP PIV

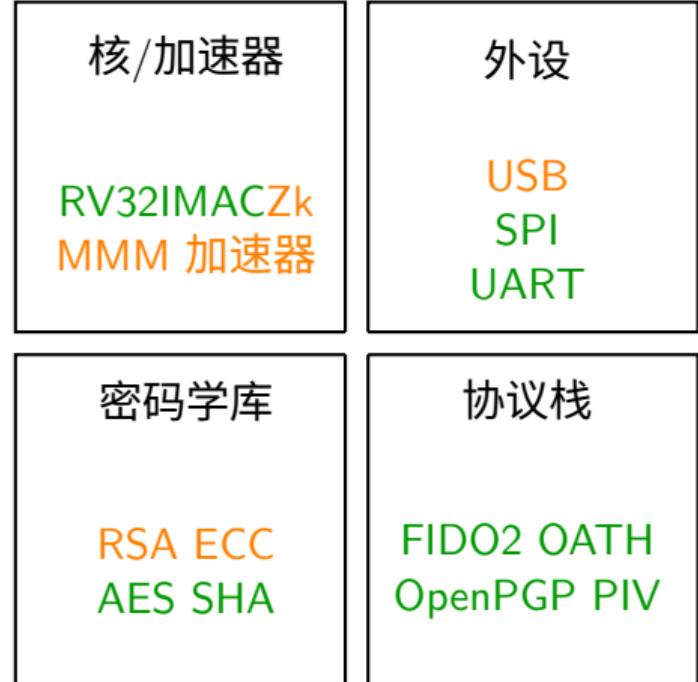
还差那么……一点点

- 一些观众：这不是随便抓一些项目来就能跑通
- 我们来假想一下要选哪些
 - 核：rocket chip, cva6, ibex...
 - 加速器：并不完全符合需求
 - 外设：随便找一些开源外设



还差那么……一点点

- 一些观众：这不是随便抓一些项目来就能跑通
- 我们来假想一下要选哪些
 - 核：rocket chip, cva6, ibex...
 - 加速器：并不完全符合需求
 - 外设：随便找一些开源外设
 - 密码学库：MbedTLS/MbedTLS，无 RSA/ECC 硬件加速



还差那么……一点点

- 一些观众：这不是随便抓一些项目来就能跑通
- 我们来假想一下要选哪些
 - 核：rocket chip, cva6, ibex...
 - 加速器：并不完全符合需求
 - 外设：随便找一些开源外设
 - 密码学库：MbedTLS/MbedTLS，无 RSA/ECC 硬件加速
 - 协议栈：Yubikey-neo, NitroKey, Gnu...

核/加速器	外设
RV32IMACZk MMM 加速器	USB SPI UART
密码学库	协议栈
RSA ECC AES SHA	FIDO2 OATH OpenPGP PIV

还差那么……一点点

- 一些观众：这不是随便抓一些项目来就能跑通
- 我们来假想一下要选哪些
 - 核：rocket chip, cva6, ibex...
 - 加速器：并不完全符合需求
 - 外设：随便找一些开源外设
 - 密码学库：MbedTLS/MbedTLS，无 RSA/ECC 硬件加速
 - 协议栈：Yubikey-neo, NitroKey, Gnu...
- 还差那么……一点点

核/加速器	外设
RV32IMACZk MMM 加速器	USB SPI UART
密码学库	协议栈
RSA ECC AES SHA	FIDO2 OATH OpenPGP PIV

我们的选择

■ 基础选择

- rocket-chip (刘玖阳是现有维护者)
- rocket-chip-blocks (旧称 sifive-blocks)
的 SPI/UART 外设
- canokey-core (党凡是主要作者)

核/加速器

rocket-chip

RV32IMACZk
MMM 加速器

外设

blocks

USB

SPI

UART

密码学库

RSA ECC
AES SHA

协议栈

canokey-core

FIDO2 OATH
OpenPGP PIV

我们的选择

■ 基础选择

- rocket-chip (刘玖阳是现有维护者)
- rocket-chip-blocks (旧称 sifive-blocks)
的 SPI/UART 外设
- canokey-core (党凡是主要作者)

■ 我们的工作

- 为 rocket-chip 增加标量密码学扩展 Zk
的支持
- 公钥密码学蒙哥马利模乘加速器
- 与 Zk、模乘加速器相对应的密码学库
- USB 1.1 外设
- 完整系统搭建

核/加速器

rocket-chip

RV32IMACZk
MMM 加速器

外设

blocks

USB

SPI

UART

密码学库

RSA ECC
AES SHA

协议栈

canokey-core

FIDO2 OATH
OpenPGP PIV

4

我们的工作

4.1

实现 RISC-V 标量密码学扩展

什么是 RISC-V 标量密码学扩展

- 在 2021 年秋，RISC-V 指令集标准化了标量密码学扩展
- 该扩展覆盖了常用的密码学操作，目的在于以较小的面积代价提供可用的加速能力

什么是 RISC-V 标量密码学扩展

- 在 2021 年秋，RISC-V 指令集标准化了标量密码学扩展
- 该扩展覆盖了常用的密码学操作，目的在于以较小的面积代价提供可用的加速能力
- 包括以下几个部分
 - 密码学位操作指令 Zbk* (Zbk_b, Zbk_c, Zbk_x)
 - NIST 密码学套件 Zkn，拥有 AES (Zknd/Zkne) 与 SHA (Zknh) 的加速能力
 - 商密密码学套件 Zks，拥有 SM4 (Zksed) 与 SM3 (Zksh) 的加速能力
 - 熵源扩展 Zkr
 - 固定时间执行延迟扩展 Zkt

为什么要实现

- 「密码学硬件密钥」虽然专注于公钥密码学，它也需要 RISC-V 标量密码学中覆盖的算法
 - FIDO2 协议需要 AES 加密算法
 - 椭圆曲线密码学算法 Ed25519 在签名过程中需要 SHA512 哈希算法

为什么要实现

- 「密码学硬件密钥」虽然专注于公钥密码学，它也需要 RISC-V 标量密码学中覆盖的算法
 - FIDO2 协议需要 AES 加密算法
 - 椭圆曲线密码学算法 Ed25519 在签名过程中需要 SHA512 哈希算法
- 在今年年初，开源社区中较为缺乏标量密码学扩展的公开实现
 - 特别的，rocket-chip 中缺乏该扩展的支持

为什么要实现

- 「密码学硬件密钥」虽然专注于公钥密码学，它也需要 RISC-V 标量密码学中覆盖的算法
 - FIDO2 协议需要 AES 加密算法
 - 椭圆曲线密码学算法 Ed25519 在签名过程中需要 SHA512 哈希算法
- 在今年年初，开源社区中较为缺乏标量密码学扩展的公开实现
 - 特别的，rocket-chip 中缺乏该扩展的支持
- 我决定为 rocket-chip 增加该扩展的支持
 - 既作为该项目的一部分
 - 也作为我的本科毕业设计（指导老师高鸣宇）¹

¹<https://blog.zenithal.me/2022/06/25/本科毕业论文-基于-Chisel-实现-RISC-V-指令集标量密码学扩展/>

标量密码学扩展的设计思路探究

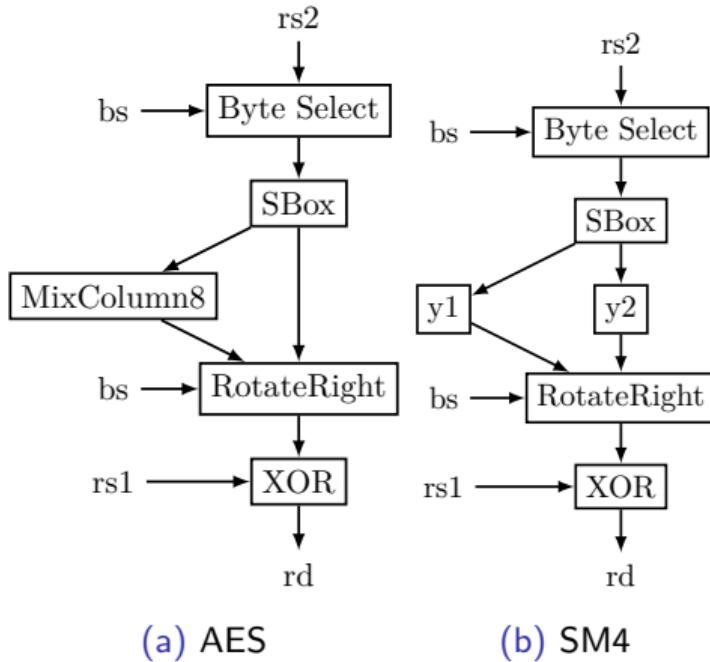
- 目的在于以「较小的面积代价」提供可用的加速能力
- 设想一种设计方法：单个指令执行全部操作
 - 例如 AES 的密钥生成和多轮加密由单个指令完成
 - `aes128_enc cipher, plain, ukey`

标量密码学扩展的设计思路探究

- 目的在于以「较小的面积代价」提供可用的加速能力
- 设想一种设计方法：单个指令执行全部操作
 - 例如 AES 的密钥生成和多轮加密由单个指令完成
 - aes128_enc cipher, plain, ukey
- 问题：这不 RISC！
- 单个指令的运算过于复杂
- 不能复用已有的指令，复用已有的面积
 - 例如异或指令与访存指令
 - 难以复用已有的运算和访存单元

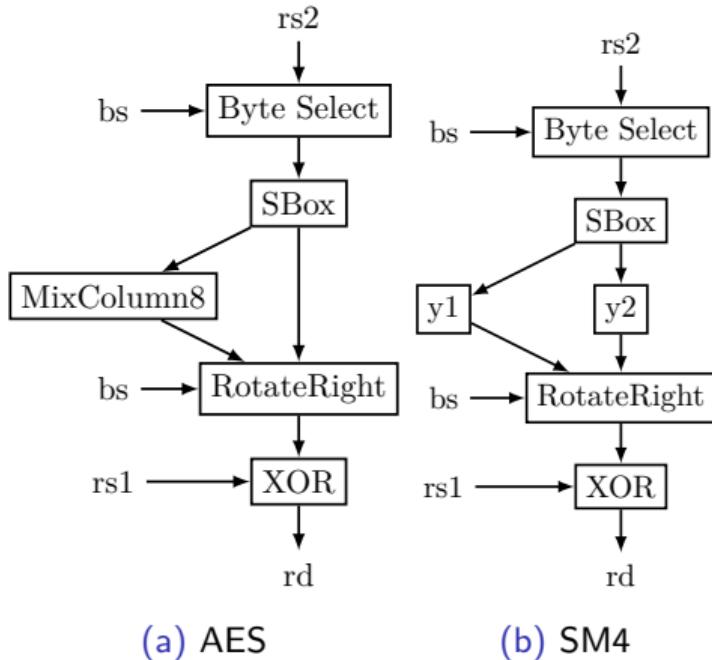
标量密码学扩展的设计思路

- 单个加速指令只做较少的特殊运算，例如 SBox 替换
- 其余运算由通用指令完成（异或、访存）
- 各个加速指令之间较为相似，能极大复用数据通路，减少面积

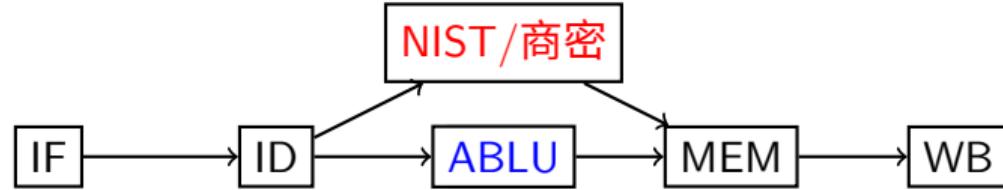


标量密码学扩展的设计思路

- 单个加速指令只做较少的特殊运算，例如 SBox 替换
- 其余运算由通用指令完成（异或、访存）
- 各个加速指令之间较为相似，能极大复用数据通路，减少面积
- 如右图所示
 - aes32{e,d}s{m,i} 四个指令可以共用数据通路
 - sm4ed/sm4ks 两个指令可以共用数据通路
 - 更进一步，这六个指令可以共用数据通路



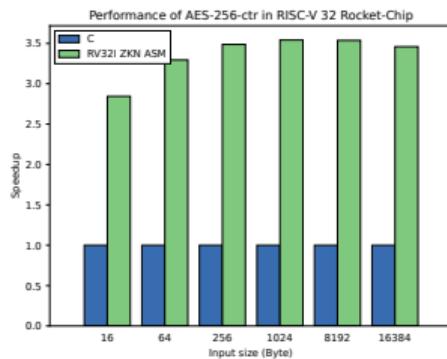
架构总览：五级流水线



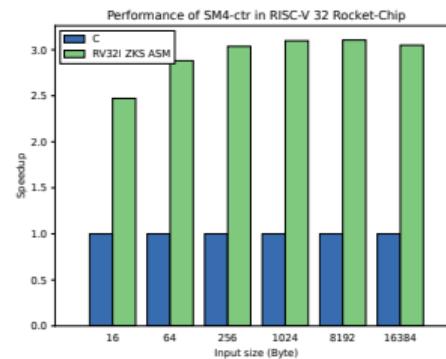
- 经典五级流水线：取指，译码，执行，访存，写回
- 我的工作：在执行阶段
 - 增加了 NIST/商密密码单元
 - 将 ALU 替换为 ABLU（算术位运算逻辑单元）
- 具体实现参考毕业论文

加速结果

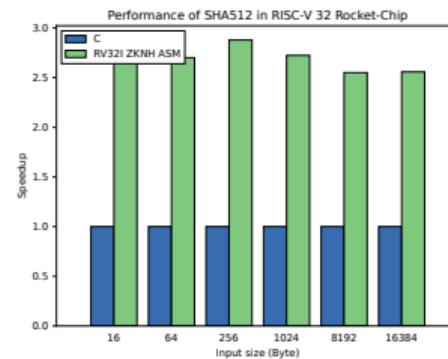
- 为了测试加速效果，给 OpenSSL 贡献了 AES, SM4, SM3 加速实现²
- 2 到 4 倍的加速比
- 密码单元的面积代价与一个乘法器相似



(a) AES-256-CTR



(b) SM4-CTR



(c) SHA512

²#18197, #18267, #18275, #18285, #18287, #18289, #18290, #18308, #18309

4.2

蒙哥马利模乘加速器

公钥密码学的运算

- 以 RSA4096 为例
- 密钥生成
 - 选取两个 2048 位的素数 p 与 q
 - 计算 4096 位整数 $n = pq$
 - 计算 $\Phi(n) = (p - 1)(q - 1)$
 - 选择 $e = 65537$ (经典值)
 - 计算 $d = e^{-1} \pmod{\Phi(n)}$
- 私钥 (d, n) , 公钥 (e, n)

公钥密码学的运算

- 以 RSA4096 为例

- 密钥生成

- 选取两个 2048 位的素数 p 与 q
- 计算 4096 位整数 $n = pq$
- 计算 $\Phi(n) = (p - 1)(q - 1)$
- 选择 $e = 65537$ (经典值)
- 计算 $d = e^{-1} \pmod{\Phi(n)}$

- 私钥 (d, n) , 公钥 (e, n)

- 签名

- 消息 m
- 私钥签名 $c = m^d \pmod{n}$
- 验证签名 $c^e \equiv m^{de} \equiv m \pmod{n}$

公钥密码学的运算

- 以 RSA4096 为例
- 密钥生成
 - 选取两个 2048 位的素数 p 与 q
 - 计算 4096 位整数 $n = pq$
 - 计算 $\Phi(n) = (p - 1)(q - 1)$
 - 选择 $e = 65537$ (经典值)
 - 计算 $d = e^{-1} \pmod{\Phi(n)}$
- 私钥 (d, n) , 公钥 (e, n)
- 签名
 - 消息 m
 - 私钥签名 $c = m^d \pmod{n}$
 - 验证签名 $c^e \equiv m^{de} \equiv m \pmod{n}$
- 核心运算为 4096 位模幂, 由一系列「模乘」组成

蒙哥马利模乘算法

- 输入 4096 位 a, b, N , 运算 $ab \pmod{N}$, 其中 N 为奇数
- 难点: 怎么对 N 取模

蒙哥马利模乘算法

- 输入 4096 位 a, b, N , 运算 $ab \pmod{N}$, 其中 N 为奇数
- 难点: 怎么对 N 取模
- 思路: 与其对 N 取模, 不如对一个「好」数字取模

蒙哥马利模乘算法

- 输入 4096 位 a, b, N , 运算 $ab \pmod{N}$, 其中 N 为奇数
- 难点: 怎么对 N 取模
- 思路: 与其对 N 取模, 不如对一个「好」数字取模
- 注意到, 在模数为 2^w 时, 取模为截取最后 w 位, 除法为向右位移 w 位
- 选取 $R = 2^{4096}$

蒙哥马利模乘算法

- 输入 4096 位 a, b, N , 运算 $ab \pmod{N}$, 其中 N 为奇数
- 难点: 怎么对 N 取模
- 思路: 与其对 N 取模, 不如对一个「好」数字取模
- 注意到, 在模数为 2^w 时, 取模为截取最后 w 位, 除法为向右位移 w 位
- 选取 $R = 2^{4096}$
- 预先运算好 $\mu = -N^{-1} \pmod{R}$
- 我们先计算 $q = \mu(ab \pmod{R}) \pmod{R}$
- 再计算 $C = (P + Nq)/R$
- 注意到以上过程中运算主要为 4096 位乘法, 取模与除法没有消耗
- 可得 $C = abR^{-1} \pmod{N}$, 离 $ab \pmod{N}$ 只有一步之遥

蒙哥马利模乘加速器

- 又观察到，乘法由加法组成（教科书算法）
- 即 $ab = a_0b + a_1b \cdot 2^1 + a_2b \cdot 2^2 + \dots$
- 加速器中只需要做 4096 位加法

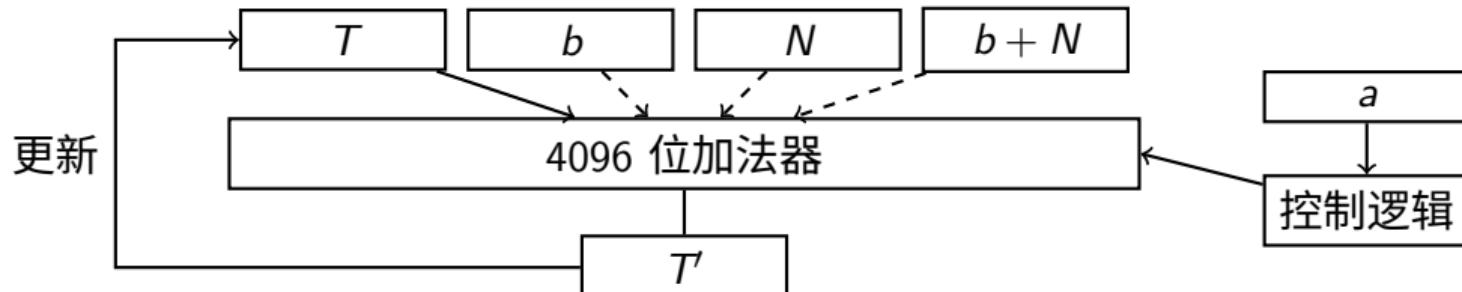


图: 加速器架构

使用 2048 位加速器运算 RSA4096

- 4096 位加法器太贵了，能不能便宜点
- 注意到 RSA4096 中， p 与 q 只有 2048 位的

使用 2048 位加速器运算 RSA4096

- 4096 位加法器太贵了，能不能便宜点
- 注意到 RSA4096 中， p 与 q 只有 2048 位的
- 中国剩余定理
 - 将 $m^d \pmod{N}$ 通过以下方式运算
 - 预先计算 $d_p = d \pmod{p}$, $d_q = d \pmod{q}$, $q_{inv} = q^{-1} \pmod{p}$ 作为私钥的一部分
 - 计算 $S_p = m^{d_p} \pmod{p}$
 - 计算 $S_q = m^{d_q} \pmod{q}$
 - 计算 $h = q_{inv} \cdot (S_p - S_q) \pmod{p}$
 - 计算 $S = S_q + h \cdot q \pmod{N}$
- 可以注意到前三步运算为 2048 位模乘

使用 2048 位加速器运算 RSA4096

- 4096 位加法器太贵了，能不能便宜点
- 注意到 RSA4096 中， p 与 q 只有 2048 位的
- 中国剩余定理
 - 将 $m^d \pmod{N}$ 通过以下方式运算
 - 预先计算 $d_p = d \pmod{p}$, $d_q = d \pmod{q}$, $q_{inv} = q^{-1} \pmod{p}$ 作为私钥的一部分
 - 计算 $S_p = m^{d_p} \pmod{p}$
 - 计算 $S_q = m^{d_q} \pmod{q}$
 - 计算 $h = q_{inv} \cdot (S_p - S_q) \pmod{p}$
 - 计算 $S = S_q + h \cdot q \pmod{N}$
- 可以注意到前三步运算为 2048 位模乘
- 最后的 4096 位模乘基于其特殊性质，也可以使用 2048 位模乘实现³

³参考党凡等人的 xRSA: <https://dang.fan/publication/icpads21-xrsa/>

可配置的加速器

- 该加速器由叶泽文用 Chisel 编写，高度可配置
- 可以较为方便地生成诸如 4096, 2048, 1024, 256 位加速器，面积与应用各有不同
- 资源充足时，可以生成 2048 位加速器以支持 RSA4096
- 资源较少时，可以生成 1024 位加速器以支持 RSA2048
- 资源匮乏时，可以生成 256 位加速器支持椭圆曲线密码学，例如 Ed25519, NIST P-256
- 较大的加速器也支持较小的算法，例如 2048 位加速器也支持 Ed25519

4.3

USB 1.1 外设

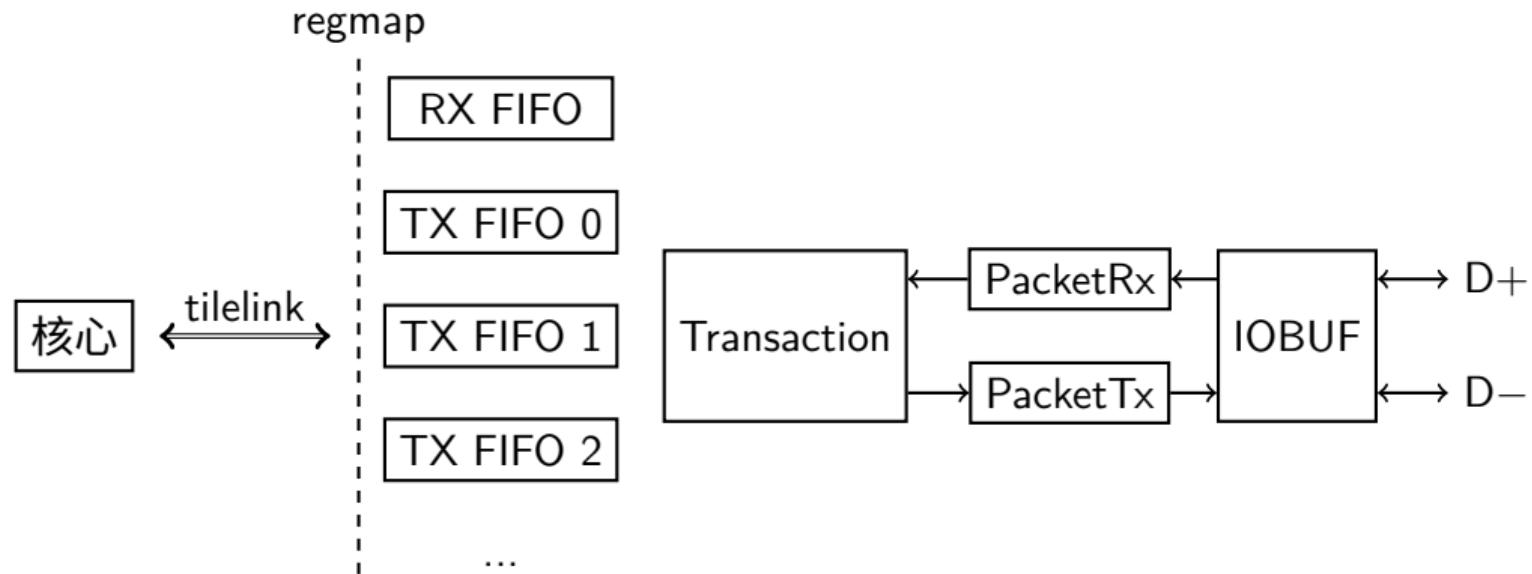
为什么要做 USB 外设

- rocket-chip-blocks 中没有相应实现
- 其余实现，并不能满足需求
 - 要么只有物理部分
 - 要么只有控制器
 - 要么难以接上总线
 - 要么需要自己写驱动以适配协议栈
- 我个人对 USB 较为感兴趣，想做个外设练练手
- 结论：自己写

USB 1.1 够用吗

- 常见误区：签名 1G 的文件，需要将文件全部传输到硬件密钥上
- 只需要传输文件的哈希以其他元数据，往往在几百字节
- USB 1.1 的 12Mbps 传输速率完全够用

USB 外设架构



最终结果

- 用 Chisel 编写，自由配置 EP 数量
- Linux 能识别

```
usb 1-2: new full-speed USB device number 119 using xhci_hcd
usb 1-2: New USB device found, idVendor=20a0, idProduct=42d4, bcdDevice= 1.00
usb 1-2: New USB device strings: Mfr=1, Product=2, SerialNumber=3
usb 1-2: Product: OpenRigil
usb 1-2: Manufacturer: canokeys.org
usb 1-2: SerialNumber: Zenithal
```

图: Linux 识别 USB 设备

4.4

完整系统搭建

硬件部分

■ 片上系统

- 单核 RV32IMACZk rocket
- 64KB D\$ 作为内存
- 2048 位蒙哥马利模乘加速器
- USB, SPI, UART (供调试)
- 288KB ROM 用于放置固件

- 片外 SPI NOR Flash, 放置私钥
- USB 通过 GPIO 引出

核/加速器
rocket-chip

RV32IMACZk
MMM 加速器

外设
blocks
USB
SPI
UART

密码学库

RSA ECC
AES SHA

协议栈
canokey-core

FIDO2 OATH
OpenPGP PIV

完整系统

软件部分

- 密码学库
 - 硬件加速 Ed25519/RSA2048/RSA4096
 - 其余密码学操作暂借自 MbedTLS
 - 之后会从 OpenSSL 移植自己写的 AES
- 协议栈：canokey-core
 - 支持 FIDO2/OATH/OpenPGP/PIV
 - 移植了 USB/SPI/UART 驱动

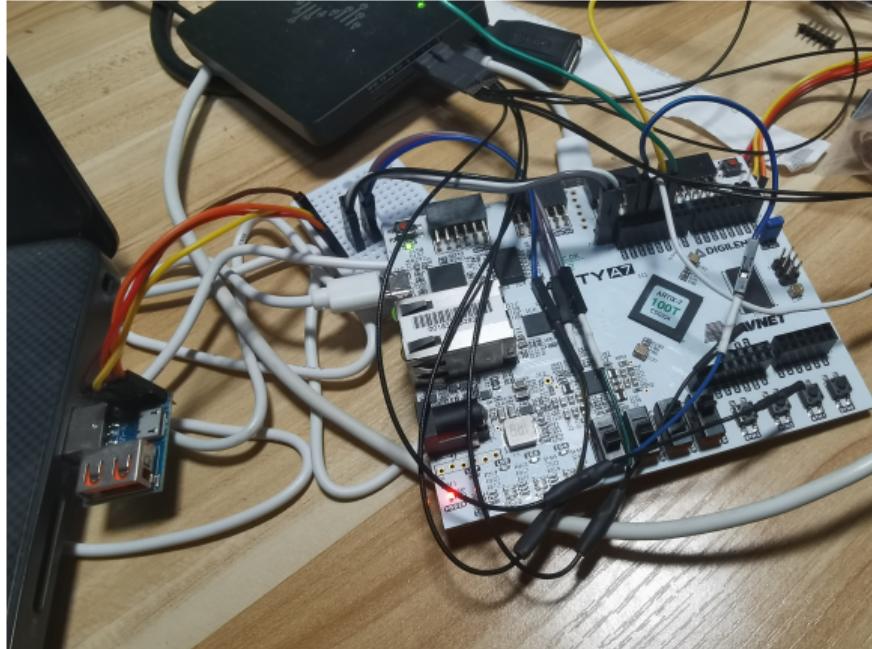


5

FPGA 原型与实验

FPGA 原型

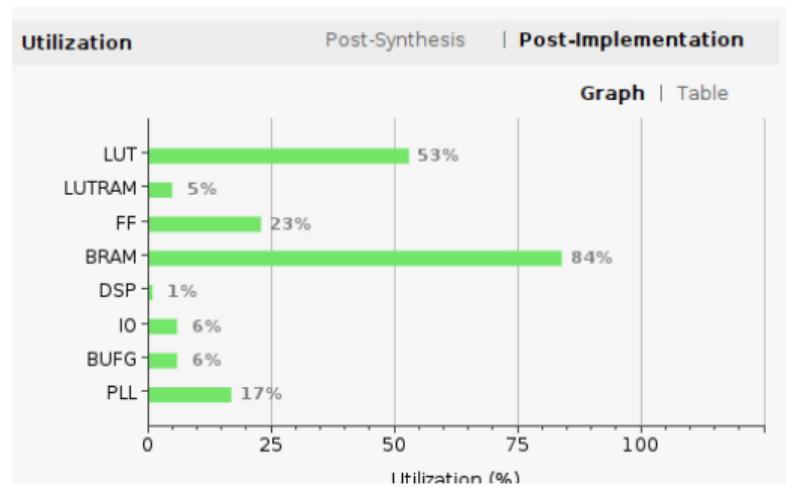
- 开发板：Arty A7-100T
- 主要特点：拥有大量 GPIO
- 关于 USB 的物理条件
 - 需要一个 1.5K 上拉电阻
 - 板子上部分 IO 拥有上拉电阻
 - 可以直接使用杜邦线
- 一块 USB 转接板，用于杜邦线到 USB 公头的转换
- 板上有 16MB 的 SPI NOR Flash，用于放置 bitstream 以及供软件使用



图：实验现场

运行参数

- 60MHz
 - USB 1.1 12Mbps 的倍数
- 53% LUT (33886/63400)
 - 核、总线、外设 15%
 - 2048 位加速器 38%
 - 如果选择 256 位加速器，5%
- 84% BRAM
 - 64K RAM 13%
 - 288K ROM 71%



试验结果

表: 各个算法时间结果

算法	软件运算	硬件加速 (签名时间)	GPG 签名整体延迟
Ed25519	2.5s	0.1s	0.45s
RSA2048	4.4s	0.5s	1.07s
RSA4096	23s	2.63s	3.00s

6

缺陷与讨论

缺陷与声明 (Disclaimer)

■ 并不安全

- 没有做到密码学硬件「安全」密钥
- 事实上，要做到安全还需要更多工作，例如
 - 片外 Flash 的保护
 - 安全启动
 - 密码学运算的测信道问题
 - 代码是否内存安全，协议栈是否有漏洞
- 是一个展示，一个讨论的基础
- 建议：不用在生产环境上，供研究与参考

■ 缺少功能

- 密码学安全的随机数发生器 (CSPRNG)
- 我不保证能做到所以没做，这是个单独的话题
- 要真正使用，还需要接入相应的 IP

- 隔壁项目：谷歌的 OpenTitan

- 也是 RISC-V，也是开源硬件，也有密码学加速器
- 我们并不是独一家
- 完全能做到和我们一样的事情，为什么我们做了呢
- 因为我们能做

- 这个项目未来会如何

- 没有答案
- 业余慢慢做
- 在社区的反馈与支持前进