

序列密码算法电路的新型物理攻防技术研究

导师：郭 箐

学生：于泽汉

上海交通大学 微纳电子学系

2018.01.11



总览

研究背景和意义

基础知识简介

算法软硬件实现

功耗分析方案

实验结果与分析

后续研究展望

研究背景和意义

基础知识简介

算法软硬件实现

功耗分析方案

实验结果与分析

后续研究展望

祖冲之算法的历史和应用

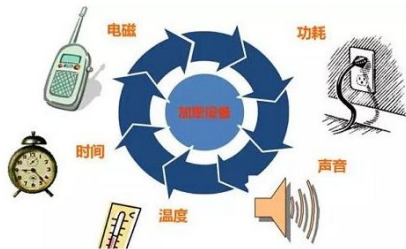


- ▶ 又称 ZUC 算法
- ▶ 我国自主研制
- ▶ 3GPP + LTE
- ▶ 序列密码算法
- ▶ 保护设备敏感信息

经过多年的学术研究和工业应用，密码学理论已经日趋系统和完善，各种密码算法广泛应用于各种工业设备，以保障系统和数据的安全。

祖冲之算法是我国第一个成为国际密码标准的密码算法，在保障 4G 通信安全中起到了重要作用。

旁路攻击对密码设备的威胁



- ▶ 理论安全 vs 实现漏洞
- ▶ 秘密信息泄露
- ▶ 功耗分析、电磁分析
- ▶ 威胁巨大

目前，那些得到广泛使用的密码算法，通常都经过数学上的严格论证，并且经过了大量专家的研究和改进，因而在理论上基本是安全的。

然而在现实中，这些算法都运行在具体设备上，因此可能会暴露出许多安全问题，攻击者可以通过各种手段获取密码设备中的秘密信息。

因此，对祖冲之算法进行旁路分析，就有助于发掘其在实际设备上的漏洞，从而提出防护方案，提高密码设备的安全性。

研究背景和意义

基础知识简介

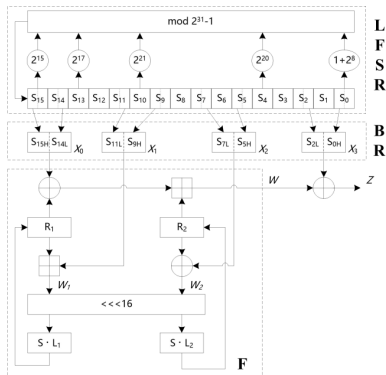
算法软硬件实现

功耗分析方案

实验结果与分析

后续研究展望

祖冲之算法的原理和流程



三层结构

- ▶ 线性反馈移位寄存器
- ▶ 比特重组
- ▶ 非线性函数

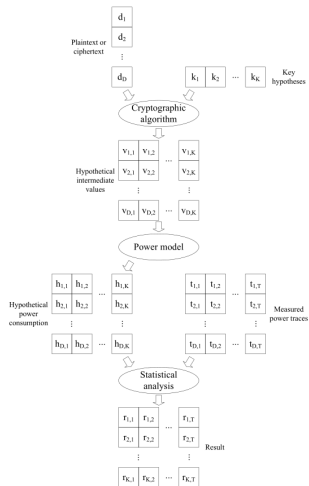
两种模式 (LFSR)

- ▶ 初始化模式
- ▶ 工作模式

两个阶段

- ▶ 初始化阶段
- ▶ 工作阶段

差分功耗分析的一般流程



- ▶ 选取合适的算法中间值位置
- ▶ 采集设备运行时的实际功耗曲线
- ▶ 根据算法计算理论中间值
- ▶ 使用合适的功耗模型将理论中间值转换为假设功耗值
- ▶ 分析假设功耗值和实际功耗曲线，挖掘所需的信息

研究背景和意义

基础知识简介

算法软硬件实现

功耗分析方案

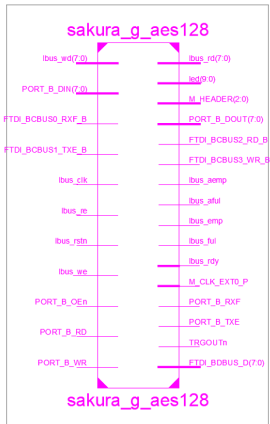
实验结果与分析

后续研究展望

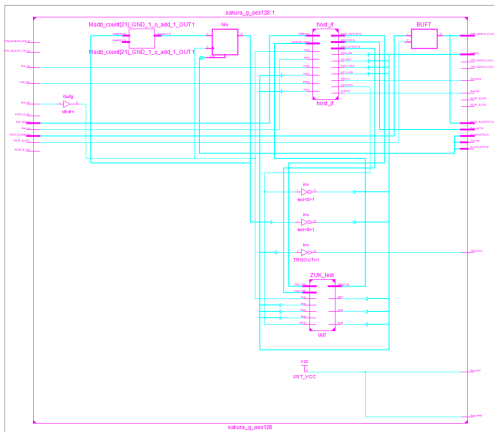
祖冲之算法的硬件实现

硬件设计软件: ISE 14.3

FPGA 型号: XC6SLX75-2CSG484



电路的输入和输出端口



电路的内部结构

祖冲之算法的软件实现

编程语言: Python 3.6
运行平台: Windows 10

```
1 def lfsrInit():
2     global k_hex, k, v_hex, v, d, s, w
3     shift_bits_list = [15, 17, 21, 20, 8, 0]
4     shift_index_list = [15, 13, 10, 4, 0, 0]
5     xv = [0] * 31
6     for i in range(0, len(shift_bits_list)):
7         s_i_shifted = circShiftLeft(s[shift_index_list[i]], shift_bits_list[i])
8         xv = modAdd_2e31m1(xv, s_i_shifted)
9         s[16] = modAdd_2e31m1(shiftLeft(w, -1), xv) # The only difference of lfsrWork() and
10            lfsrInit()
11     if s[16] == [0]*31:
12         s[16] = [1]*31
13     for i in range(0,16):
14         s[i] = s[i+1]
15
16 def lfsrWork():
17     global k_hex, k, v_hex, v, d, s
18     shift_bits_list = [15, 17, 21, 20, 8, 0]
19     shift_index_list = [15, 13, 10, 4, 0, 0]
20     xv = [0] * 31
21     for i in range(0, len(shift_bits_list)):
22         s_i_shifted = circShiftLeft(s[shift_index_list[i]], shift_bits_list[i])
23         xv = modAdd_2e31m1(xv, s_i_shifted)
24         s[16] = xv # The only difference of lfsrWork() and lfsrInit()
25     if s[16] == [0]*31:
26         s[16] = [1]*31
27     for i in range(0,16):
28         s[i] = s[i+1]
```

线性反馈移位寄存器模块

```
1 def bitReorganization():
2     global x, s
3     x[0] = s[15][0:16] + s[14][:-16:]
4     x[1] = s[13][:-16:] + s[9][0:16]
5     x[2] = s[7][:-16:] + s[5][0:16]
6     x[3] = s[2][:-16:] + s[0][0:16]
```

比特重组模块

```
1 def nonLinearFunction():
2     global w, x, r1, r2
3     w = binaryAdd(binaryXor(x[0], r1), r2)
4     w1 = binaryAdd(r1, x[1])
5     w2 = binaryXor(r2, x[2])
6     r1 = sbboxFzuc(linearTransform(w1[-16:]+w2[0:16], 1))
7     r2 = sbboxFzuc(linearTransform(w2[-16:]+w1[0:16], 2))
```

非线性函数模块

祖冲之算法的软件实现

研究背景和意义

基础知识简介

算法软硬件实现

功耗分析方案

实验结果与分析

后续研究展望

研究背景和意义

基础知识简介

算法软硬件实现

功耗分析方案

实验结果与分析

后续研究展望

研究背景和意义

基础知识简介

算法软硬件实现

功耗分析方案

实验结果与分析

后续研究展望

谢谢！