I, for One, Welcome Our New Power Analysis Overlords

aka: ChipWhisperer-Lint

by Colin O’Flynn
Something About Me.

• Assistant Professor ➔ Dalhousie University
  • Very recent change – if you are interested in doing a MASc/PhD in this work and living in Halifax, NS, Canada while so please talk to me!

• C.T.O. ➔ NewAE Technology Inc.
  • Startup on embedded security, we make test equipment, run trainings, do open-source stuff!
AES Hardware Acceleration?

SAM L10 and L11 Microcontroller Family

Industry’s Lowest Power 32-bit MCUs, First to Offer Chip-Level Arm® TrustZone® Technology

With the increasing growth of IoT and points and, consequently, the increased frequency of security power consumption while adding robust security. The SAM L10 and L11 MCU family takes an innovative

- AES Hardware Acceleration?

Robust Security

SAM L10 and L11 MCU’s integrates ARM® TrustZone® techs from both the physical

- AES Hardware Acceleration?

STM32F405/41

The STM32F405/415 lines are designed for medical, industrial, and power efficiency, embedded memories, and rich peripherals. The STMicroelectronics’ STM32F405/415 offers the full performance of the

- AES Hardware Acceleration?

STM1318: Using the XMEGA built accelerator

Features

- Full compliance with AES (FIPS Publication 197, 2002)
- Both encryption and decryption procedures
- 128-bit Key and State memory
- XOR operation to state memory useful for cipher block coding
- Sequential access to State and Key memories
- Optional Interrupt- and DMA request on AES complete

1 Introduction

The XMEGA™ AES Crypto Module supports the Advanced Encryption Standard (AES), and can perform encryption and decryption. The module supports a key length of 128 bits. The 128-bit key block and 128-bit data block (pl enforced ciphertext) must be loaded into the Key and State memory in the XMEGA Crypto Module. The AES uses 163-bit nodes in a tree of one encryption/desecrypting.

Kinetis K24F Sub-Family Data Sheet

120 MHz ARM Cortex-M4-based Microcontroller with FPU

The K24 product family members are optimized for cost-sensitive applications requiring low-power, USB connectivity, and up to 256 K of embedded SRAM. These devices share the comprehensive enhancement and scalability of the K24.

This product offers:

- Run power consumption down to 250 μA/MHz, Static power consumption down to 5.8 μA with full state retention and 5 μA wakeup. Lowest static mode down to 359 μA
- USB L5/F5 OTG 2.0 with embedded 3.3 V, 120 mA LDO
- Vreg, with USB device crystal-less operation

Communication interfaces

- USB full-speed/low-speed On-The-Go controller
- Controller Area Network (CAN) module
- Three SPI modules
- Three I2C modules
- Three UARTs
- Secure Digital Host Controller (SDHC)
- I2S module

Fieldbus

- Two 8-channel FlexTimers (PWM/rt timer)
- Two 2-channel FlexTimers (PWM/Quad I/O)
- 32-bit P ITS and 16-bit low-power timers
- Real-time clock
- Programmable delay block

Clock

- 3 to 50 MHz and 32 kHz crystal oscillator
- PLL, PLL, multi-channel internal oscillators
- 48 MHz Internal Reference Clock (IRC48M)

Operating Characteristics
Side Channel Power Analysis
Data Busses...
Model of power consumption used with guess of secret information. Only ONE guess should match “real” measurement assuming our model is good/accurate.

<table>
<thead>
<tr>
<th>Input Data</th>
<th>Power Measurement</th>
<th>Output of Leakage Model Byte Guess = 0x00</th>
<th>Output of Leakage Model Byte Guess = 0x01</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xC7</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x1F</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x2C</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x89</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x01</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xD2</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Using in Real Life

Why Light Bulbs May Be the Next Hacker Target

By JOHN MAUGHIN  NOV. 2, 2016

Packet 41 (first 16-byte packet) Processing using AES-CBC

Power Ratio

The Internet of Things, activated through apps, promises tremendous conveniences to homeowners. But it may also prove irresistible to hackers. | Courtesy Photographs by Ben Shapero

SAN FRANCISCO — The so-called Internet of Things, its proponents argue, offers many benefits: energy efficiency, technology so convenient it can anticipate what you want, even reduced congestion on the roads.

Now here’s the bad news: Putting a bunch of wirelessly connected devices in one area could prove irresistible to hackers. And it could allow them to spread malicious code through the air, like a flu virus on an airplane.

Researchers report in a paper to be made public on Thursday that they have uncovered a flaw in a wireless technology that is often included in smart

On the Power of Power Analysis in the Real World: A Complete Break of the KeeLoq Code Hopping Scheme

Thomas Eisenbarth1, Timo Kasper1, Amir Moradi1,2, Christoph Paar1, Mahmod Salami-Meida1,2, and Mohammad T. Mahoori Shamsi3

1 Heisei Institute for IT Security
Ruhr University Bochum, Germany
2 Department of Computer Engineering and Electronic Research Center
Sharif University of Technology, Tehran, Iran
3 eisenbarth,tkasper,amirid,cpearl@crypto.rub.de
salami,mahoori@barfu.edu

Abstract. KeeLoq remote keyless entry systems are widely used for access control purposes such as garage openers or car door systems. We present the first successful differential power analysis attacks on

Schneier on Security

Breaking the Xilinx Virtex-II FPGA Bitstream Encryption

It’s a power analysis attack, which makes it much harder to defend against. And since the attack model is an engineer trying to reverse-engineer the chip, it’s a valid attack.

Abstract: Over the last two decades, FPGAs have become central components for many advanced digital systems, e.g., video signal processing, network routers, data acquisition and military systems. In order to protect the intellectual property and to prevent fraud, e.g., by cloning an FPGA or manipulating its content, many current FPGAs employ a bitstream encryption feature. We develop a successful attack on the bitstream encryption engine integrated in the widespread Virtex-II Pro FPGAs from Xilinx, using side-channel analysis. After measuring the power consumption of a single power-up of the device and a modest amount of offline computation, we are able to recover all three different keys used by its triple DES module. Our method allows extracting secret keys from any real-world device where the bitstream encryption feature of Virtex-II Pro is enabled. As a consequence, the target product can be cloned and manipulated at will of the attacker. Also, more advanced attacks such as reverse
Problem: What is Leakage Model?

Don’t be “too smart”, try all leakage models if you want CPA attack maybe?

But very time consuming to test lots of models… how to improve on this?
What about Machine Learning?

I tried it... once.

(Circa 2003).
What about Machine Learning?

OK twice. But I mean it was running in BASIC*, so it doesn’t count.

(Circa 2002).

*Artificial Neural Network implemented in BASCOM-AVR, which is compiled BASIC running on embedded microcontroller.
What about Machine Learning?

• Problem is very well formed for machine learning:
  • Huge datasets (we can create as many as we want).
  • SNR is something we can control/optimize very easily.
  • Relatively easy to define outcome.

• Problems with machine learning:
  • I haven’t looked at it since 2003 with my sweet ANN robot.
  • We have a pretty good idea of underlying models, and we’d like to validate specific ones.
Dumb Method: Step #1

```python
class AES128_Leakage(Leakage_Base):
    # Implements AES-128, saving internal states in a dictionary for analysis
    name = 'AES128'
    ks = None

    leakage_points = [
        'Plaintext',
        'Key',
        'Round 0: AddRoundKey Output',
        'Round 1: SubBytes Output',
        'Round 1: ShiftRows Output',
        'Round 1: MixColumns Output',
        'Round 1: RoundKey',
        'Round 2: AddRoundKey Output',
        'Round 2: SubBytes Output',
        'Round 2: ShiftRows Output',
        'Round 2: MixColumns Output',
        'Round 2: RoundKey',
        'Round 2: AddRoundKey Output',
        'Round 3: SubBytes Output',
        'Round 3: ShiftRows Output',
        'Round 3: MixColumns Output',
        'Round 3: RoundKey',
        'Round 3: AddRoundKey Output',
        'Round 4: SubBytes Output',
        'Round 4: ShiftRows Output',
        'Round 4: MixColumns Output',
        'Round 4: RoundKey',
        'Round 4: AddRoundKey Output',
        'Round 5: SubBytes Output',
        'Round 5: ShiftRows Output',
        'Round 5: RoundKey',
        'Round 5: AddRoundKey Output',
        'Round 5: MixColumns Output',
        'Round 5: AddRoundKey Output',
        'Round 6: SubBytes Output',
        'Round 6: ShiftRows Output',

    Nr = 10
    state = pt
    ret['Plaintext'] = self.flatten(state[:])
    ret['Key'] = self.flatten(self.ks[0])

    state = [state[1] ^ self.ks[i][1] for i in range(16)]
    ret['Round 0: AddRoundKey Output'] = self.flatten(state[:])

    for r in range(1, Nr):
        state = subbytes(state)
        ret['Round ' + str(r) + ': SubBytes Output'] = self.flatten(state[:])

        state = shiftrows(state)
        ret['Round ' + str(r) + ': ShiftRows Output'] = self.flatten(state[:])

        state = mixcolumns(state)
        ret['Round ' + str(r) + ': MixColumns Output'] = self.flatten(state[:])

        ret['Round ' + str(r) + ': RoundKey'] = self.flatten(self.ks[r])

        state = [state[1] ^ self.ks[r][1] for i in range(16)]
        ret['Round ' + str(r) + ': AddRoundKey Output'] = self.flatten(state[:])

    state = subbytes(state)
    ret['Round 10: SubBytes Output'] = self.flatten(state[:])
```

DO ALL THE THINGS!!!
Dumb Method: Step #2

Amazon EC2 Pricing

With On-Demand instances you only pay for EC2 instances you use. The use of On-Demand instances frees you from the costs and complexities of planning, purchasing, and maintaining hardware and transforms what are commonly large fixed costs into much smaller variable costs.

<table>
<thead>
<tr>
<th>Compute Optimized - Current Generation</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>c5.large</td>
<td>2</td>
<td>9</td>
<td>4 GB</td>
<td>EBS Only</td>
</tr>
<tr>
<td>c5.xlarge</td>
<td>4</td>
<td>17</td>
<td>8 GB</td>
<td>EBS Only</td>
</tr>
<tr>
<td>c5.2xlarge</td>
<td>8</td>
<td>34</td>
<td>16 GB</td>
<td>EBS Only</td>
</tr>
<tr>
<td>c5.4xlarge</td>
<td>16</td>
<td>68</td>
<td>32 GB</td>
<td>EBS Only</td>
</tr>
<tr>
<td>c5.9xlarge</td>
<td>36</td>
<td>141</td>
<td>72 GB</td>
<td>EBS Only</td>
</tr>
<tr>
<td>c5.16xlarge</td>
<td>72</td>
<td>281</td>
<td>144 GB</td>
<td>EBS Only</td>
</tr>
<tr>
<td>c5dlarge</td>
<td>2</td>
<td>9</td>
<td>4 GB</td>
<td>1 x 50 NVMe SSD</td>
</tr>
<tr>
<td>c5dxlarge</td>
<td>4</td>
<td>17</td>
<td>8 GB</td>
<td>1 x 100 NVMe SSD</td>
</tr>
<tr>
<td>c5d.2xlarge</td>
<td>8</td>
<td>34</td>
<td>16 GB</td>
<td>1 x 200 NVMe SSD</td>
</tr>
<tr>
<td>c5d.4xlarge</td>
<td>16</td>
<td>68</td>
<td>32 GB</td>
<td>1 x 400 NVMe SSD</td>
</tr>
<tr>
<td>c5d.9xlarge</td>
<td>36</td>
<td>141</td>
<td>72 GB</td>
<td>1 x 900 NVMe SSD</td>
</tr>
<tr>
<td>c5d.16xlarge</td>
<td>72</td>
<td>281</td>
<td>144 GB</td>
<td>1 x 1800 NVMe SSD</td>
</tr>
<tr>
<td>c4.large</td>
<td>2</td>
<td>8</td>
<td>3.75 GB</td>
<td>EBS Only</td>
</tr>
<tr>
<td>c4.xlarge</td>
<td>4</td>
<td>16</td>
<td>7.5 GB</td>
<td>EBS Only</td>
</tr>
<tr>
<td>c4.2xlarge</td>
<td>8</td>
<td>31</td>
<td>15 GB</td>
<td>EBS Only</td>
</tr>
<tr>
<td>c4.4xlarge</td>
<td>16</td>
<td>62</td>
<td>30 GB</td>
<td>EBS Only</td>
</tr>
<tr>
<td>c4.8xlarge</td>
<td>36</td>
<td>132</td>
<td>60 GB</td>
<td>EBS Only</td>
</tr>
</tbody>
</table>
How do you known she is a witch?

Remember these?

Most micros not specifically designed as secure element don’t talk about power analysis.

But maybe you actually care about side-channel resistance?

Not so much you are going to spend $$$ getting a validated crypto core.

What to do???
Test Vector Leakage Assessment (TVLA)

• Based on proposal “A testing methodology for sidechannel resistance validation” by Gilbert Goodwill, Benjamin Jun, Josh Jaffe, and Pankaj Rohatgi of Cryptography Research Inc.
Welch’s t-test

You have observations of two random variables (the test situations). How confident are you they have equal means?

If they had un-equal means, you could form a probability that a given measurement (observation of encryption) came from one set vs. the other.

\[
t = \frac{\overline{X}_1 - \overline{X}_2}{\sqrt{\frac{s_1^2}{N_1} + \frac{s_2^2}{N_2}}}
\]
T-Test Results & Expansions

Test 1: Test 0: Fixed/Random Plaintext
Maximum t: 48.228980 @ 657 [FAIL]

Test 65: HW: Round 5: SubBytes Output byte 0
Maximum t: 4.998180 @ 1022 [FAIL]
GET ON
WITH IT
ChipWhisperer-Lint Objectives

• Provide method of running pass/fail security tests.
• Provide information regarding specific leakage model in use.
• Make simple method of automatically running these tests.
Examples of Hardware Crypto
Examples: Running on some Micros

Chosen five “representative” samples of micros you might use for various tasks. Specifically:

• Mix of application-specific, general-purpose.

• NONE of these are “secure” devices marked for credit cards, content delivery, etc.

Disclosure:

• This is NOT an “attack” specific to the micros – so nothing is more broken now than it was a week ago. Rather this is a generic weakness across ALL similar devices. The “attack” only exists when people mis-use the crypto.

• Vendors have been contacted or I’ve attempted to contact them (only last week as I was running late, sorry about that vendors).
Target Devices

- ST STM32F415 (Arm Cortex-M4 Core)
- NXP Kinetis K24 (Arm Cortex-M4 Core)
- Espressif ESP32 (Tensilica Xtensa LX6 Core)
- Atmel SAM4L (Arm Cortex-M4 Core)
General Capture Setup
STM32F415 – Hardware Crypto

- Hardware AES engine.
- Relatively powerful Arm Cortex M4 core at 168 MHz.
- Lots of I/O (USB, CAN, etc).

![STM32F415xx Diagram](image)

Features:
- Core: ARM® 32-bit Cortex®-M4 CPU with FPU, Adaptive real-time accelerator (ART Accelerator™) allowing Dual state execution from Flash memory, frequency up to 168 MHz, memory protection unit, 216 DMIPS/1.25 DMIPS/MHz (Dhrystone 2.1), and DSP instructions
- Memories
  - Up to 1 Mbyte of Flash memory
  - Up to 192+4 Kbytes of SRAM including 64-Kbyte of DCI (data coupled memory) data RAM
- Flexible static memory controller supporting Compact Flash, SRAM, PSRAM, NOR and NAND memories
- LCD parallel interface, 8080/8052 modes
- Clock, reset and supply management
  - 1.8 V to 3.6 V application supply and I/Os
  - POR, PDR, PVD and BOR
  - 4-to-26 MHz crystal oscillator
  - Internal 16 MHz factory-trimmed RC (1% accuracy)
  - 32 kHz oscillator for RTC with calibration
  - Internal 32 kHz RC with calibration
- Low-power operation
  - Standby, Stop and Standby modes
- ARM Cortex-M4 32b MCU+FPU, 210DMIPS, up to 1MB Flash/192+4KB RAM, crypto, USB OTG HS/FS, Ethernet, 17 TIMs, 3 ADCs, 15 comm. interfaces & camera
- Datasheet – production data
- Features
  - Cortex-M4 Embedded Trace Macrocell™
  - Up to 140 I/O ports with interrupt capability
  - Up to 136 fast I/Os up to 84 MHz
  - Up to 138 5-V-tolerant I/Os
  - Up to 15 communication interfaces
    - Up to 3 x SCI interfaces (SMBus/PMBus)
    - Up to 4 USARTs/2 UARTs (10.5 Mbps, ISO 7816 interface, LIN, IrDA, modem control)
    - Up to 3 SPIs (42 Mbps/8), 2 with muxed full-duplex P2P to achieve audio class accuracy via internal audio PUL or external
Test 88: HD: Round 0: AddRoundKey Output to Round 1: AddRoundKey Output

Maximum t: 6.304197 @ 176 [FAIL]

AES State to State Hamming Distance Leakage
STM32F415 Summary

- Standard AES State to State Register Leakage (Hamming distance)
- Narrow window needed around leaking point (here around point 205 – this is last round attack)
- Break is ~6K traces
Kinetis K24

- 120 MHz Cortex-M4
- Hardware AES peripheral (also DES, SHA)
Test 127: HD: Round 1: SubBytes Output to Round 1: AddRoundKey Output

Maximum t: 7.205168 @ 722 [FAIL]

Hamming Distance leakage on S-Box input to output
K24F Summary

• Interesting Sbox Input to Output Leakage
• Appears to process in 32-bit sections, possibly T-Table or other implementation.
• Break is ~14K traces.
BONUS: Catching library leakages (either in examples or even ROM code)

Test 81: HD: Key to Round 10: SubBytes Output

Maximum t: 40 174129 @ 1998 [FAIL]
Espressif ESP32

1.6 Block Diagram

- Single or dual core device.
- Super-low cost device targets IoT.
- AES accelerator, also used to encrypt external SPI flash.
Test 1: Test 0: Fixed/Random Plaintext

Maximum t: 48.228980 @ 637 [FAIL]
ESP32 Summary

• AES-128 hardware accelerator is leaky, breakable with CPA.

• Leakage Model: HW S-Box Output, approx. 18K traces needed.

• Briefly looked at usage during boot – not immediately successful. More research needed to understand when decryption core is used.
Atmel Microchip SAM4L

Summary

Atmel's SAM4L series is a member of a family of Flash microcontrollers based on the high performance 32-bit ARM Cortex-M4 RISC processor running at frequencies up to 48 MHz.

The SAM4L series embeds state-of-the-art picopower technology for ultra-low power consumption. Combined power control techniques are used to bring active current consumption down to 50 μA/MHz. The device allows a wide range of options between functionality and power consumption, giving the user the ability to reach the lowest possible power consumption with the feature set required for the application. The WAIT and RETENTION modes provide full logic and RAM retention, associated with fast wake-up capability (<1.5 ms) and a very low consumption of, respectively, 3 μA and 1.5 μA. In addition, WAIT mode supports SleepWalking features. In BACKUP modes, CPU peripherals and RAM are powered off and, while consuming less than 0.9 μA with external interrupt wake-up supported.

The SAM4L series offers a wide range of peripherals such as segment LCD controller, embedded hardware capacitive touch (QTouch), USB device & embedded host, 128-bit AES and audio interfaces in addition to high speed serial peripherals such as USART, SPI and I²C. Additionally the Peripheral Event System and SleepWalking allows the peripherals to communicate directly with each other and make intelligent decisions and decide to wake-up the system on a qualified events on a peripheral level; such as I²C address match or and ADC threshold.

Features

- Core
  - ARM® Cortex™-M4 running at up to 48 MHz
  - Memory Protection Unit (MPU)
  - Thumb®-2 instruction set
- picopower® Technology for Ultra-low Power Consumption
  - Active mode down to 50 μA/MHz with configurable voltage scaling
  - High performance and efficiency: 28 coremark/MHz
  - Deep sleep mode down to 3 μA with fast wake-up times (<1.5 ms) supporting SleepWalking

• Hardware AES Engine…. With “countermeasures”
• Very low-power, Arm Cortex M4 @ 48 MHz.
• USB support.
Countermeasures On (incorrectly configured)

Test 55: HD: Round 2: AddRoundKey Output to Round 3: AddRoundKey Output

Maximum t: 5.310029 @ 141 [FAIL]

Round to round leakage

T-Test maximum
Bonus: Detecting Datasheet Bugs

• Datasheet is missing critical configuration information that results in countermeasures not being effective.
• Based on feedback from Atmel these changes were made to software and tests re-run.

Specific error: DRNGSEED MUST be written as part of initialization. Datasheet only references setting this after changing key size (default is 128-bit so by default no change occurs), and otherwise claims countermeasures are always on unless explicitly disabled.
Atmel Microchip SAM4L Summary

• Some of the countermeasures ineffective (‘add noise’ does not really do anything for example).

• CPA attack effective in ~3000 traces.

• Jitter stops CPA from working.
  • Simple resync insufficient to fix → either random state included or more work needed.
## Summary

<table>
<thead>
<tr>
<th>Device</th>
<th>AES-128 Cycles*</th>
<th>Target Leakage</th>
<th>CPA Traces</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>STM32F415</td>
<td>493</td>
<td>Round to Round HD</td>
<td>~6 000</td>
<td></td>
</tr>
<tr>
<td>Kinetis K24F</td>
<td>475</td>
<td>S-Box Input to Output HD</td>
<td>~14 000</td>
<td></td>
</tr>
<tr>
<td>ESP32</td>
<td>252</td>
<td>S-Box Output HW</td>
<td>~18 000</td>
<td>Calling ROM function directly.</td>
</tr>
<tr>
<td>SAM4L</td>
<td>81</td>
<td>Round to Round HD</td>
<td>~3 000</td>
<td>Jitter countermeasures DISABLED.</td>
</tr>
</tbody>
</table>

*Includes overhead of calling crypto library function + setup for single block. NOT speed-optimized.

All devices: Measurement with ChipWhisperer-Lite Capture, 7.37 MHz, 29.48 MS/s.
How to Care

• Use SECURE device if you need side-channel power analysis protection. Ensure it includes ratings based on relevant industry standard.

• Example: Common Criteria rated secure element or microcontroller.
Examples of Checking Software Libraries
Getting Started

• You’ll need to configure an example target to provide encryption.
  • Lot of examples in ChipWhisperer repo.
  • Additional examples on ChipWhisperer-Lint repo showing complete attack

ChipWhisperer Lint Detection on mbed TLS Library

This demo is designed to run AES using the mbed TLS library against a number of ARM targets. This demonstrates the power of automated analysis in determining where leakage exists, and why testing against ALL possible variants is useful for detecting leakage that might go unnoticed on specific builds/variants.

The setup involves (a) building possible variants of the library with different compilers, library options, and compiler flags, (b) running the target code on different physical architectures, and (c) performing automated side-channel analysis of the resulting power traces.

These are provided by:

(a) A special build script (would be specific to your build environment). (b) The ChipWhisperer-Capture software connected to a ChipWhisperer-Lite + UFO Board, run in a basic script. (c) The ChipWhisperer-Lint software.

(a) Build Process

The target is the mbed TLS library. The automated test script performs the following actions:

1. Perform "git pull" to get latest code from public repository.
2. For each variant:
   ◦ Autogenerate makefile to build binary (based on settings below).
Further Example - C.I. Test Suite

Source code is crypto library under test (for SW), but can be IP core for FPGA tests.

Various binaries generated – library is compiled for supported platforms and with various options a user might enable.

Binaries loaded onto test platform (for example, based on UFO board), but can also use existing development kits that have been instrumented to take power measurements.

Capture can be done with regular oscilloscope. Here ChipWhisperer hardware (open-source versions available) shown, which simplifies setup considerably.

Captured power traces analyzed by ChipWhisperer-Lint. Can run a local server or use more powerful cloud style server.
### Test Results

| Test Number | Test Name | Minimum Time | 10147 | 10147 | 7147 | 7143 | 9963 | 9963 | 6699 | 6699 | 8795 | 4475 | 7055 | 4071 | 11659 | 17259 | 7283 | 10147 | 9967 | 6363 | 9559 |
|-------------|-----------|--------------|-------|-------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|
| -           | Minimum Time | 10147 | 10147 | 7147 | 7143 | 9963 | 9963 | 6699 | 6699 | 8795 | 4475 | 7055 | 4071 | 11659 | 17259 | 7283 | 10147 | 9967 | 6363 | 9559 |
| -           | Maximum Time | 10147 | 10147 | 7147 | 7143 | 9963 | 9963 | 6699 | 6699 | 8795 | 4475 | 7055 | 4071 | 11659 | 17259 | 7283 | 10147 | 9967 | 6363 | 9559 |
| 3           | HD: Plaintext to Round 0: AddRoundKey Output | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 |
| 6           | HW: Key | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 |
Leakage based on certain tests may not be present on specific configurations/hardware.

Testing on a single device is insufficient, even when devices are of similar family (in this case all ARM, STM32F).
Potential timing attack on STM32F2

ROM lookup tables have non-constant time due to 128-bit wide FLASH bus (ART accelerator) on F2, but NOT present on F0/F1.

Almost impossible to catch this error without hardware validation (cycle-counting would NOT catch).
The End.
Conclusions

• Side channel power analysis – you should care. It’s real. It’s here.
• It can break most generic AES (or other crypto) implementations. ChipWhisperer-Lint can help you with leakage model if you need.
• We do automated software testing, why not also add hardware-in-the-loop automated testing to detect power side-channel flaws?

www.ChipWhisperer.com
github.compytpythohon/newaetech/chipwhisperer-lint
coflynn@newae.com