A FRAMEWORK FOR EMBEDDED HARDWARE SECURITY ANALYSIS

by

Colin O'Flynn

Submitted in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

at

Dalhousie University
Halifax, Nova Scotia
June 2017

To my family (old and new).

Table of Contents

List of Tables

List of Figures

Abstract

Embedded computers are unavoidable in our daily life, and our interaction with them only looks to increase as more products include the words 'Internet of Thing' in their selling features. Embedded computers can be found in our credit cards, in our cars, and in our thermostats. With such a wide distribution of embedded computers one might expect the companies designing and building them to look towards the large body of research present in academia about attacking and securing these devices.

But a gap exists between these two worlds, and the result can be seen in the many attacks against embedded devices presented every year at conferences such as Black Hat and DEFCON. This thesis introduces low-cost and open-source hardware and software that allows industry to more easily apply recent research publications, so this gap can be closed.

The fields of side-channel power analysis and fault injection allows us to successfully attack even strong cryptographic protocols, as these protocols can be broken when implemented on embedded devices. Understanding these attacks is critical to build strong devices that have to resist attacks for the next five to twenty years, especially where the devices may have limited ability to be updated. In addition to introducing a novel architecture for the analysis tool, this thesis includes several examples of attacks against various devices including small microcontrollers, field programmable gate arrays, embedded Linux computers, and IEEE 802.15.4 wireless nodes.

List of Abbreviations and Symbols Used

$\oplus$	X-OR Operation
K	Secret encryption key
K_j	Byte j of secret key K
σ_X	Standard deviation of X
σ_X^2	Variance of X
μ_X	Mean of X
$r(t)$	Continuous-time function of time t
$r[n]$	Discrete-time function of sample n
$\|X\|$	Absolute value of X
BW	Analog bandwidth
ADC	Analog to Digital Converter
AES	Advanced Encryption Standard
ARM	Advanced RISC Machines (type of 32-bit microcontroller)
AVR	An 8-bit RISC microcontroller (not an abbreviation)
CAN	Controller Area Network
CCM	Counter with CBC-MAC (AES Mode)
CPA	Correlation Power Analysis
COTS	Computer Off the Shelf
DCM	Digital Clock Manager (part of Xilinx FPGAs)
DES	Data Encryption Standard
DIP	Dual In-line Pin (IC packaging)
DPA	Differential Power Analysis
DSO	Digital Storage Oscilloscope
DUT	Device Under Test

EPROM	Erasable Programmable Read-Only Memory
FPGA	Field Programmable Gate Array
GND	Ground, negative supply voltage
GSR	Global Success Rate
HD	Hamming Distance
HW	Hamming Weight
IoT	Internet-of-Things
LNA	Low Noise Amplifier
MOSFET	Metal Oxide Semiconductor Field Effect Transistor
OS	Operating System
PGE	Partial Guessing Entropy
PSR	Partial Success Rate
RAM	Random Access Memory
SAD	Sum of Absolute Differences
SoC	System on a Chip
SNR	Signal to Noise Ratio
USB	Universal Serial Bus
VC	Varying Clock
VCC	Positive supply voltage

VCO	Voltage Controlled Oscillator

Acknowledgements

Pursuing a PhD was not something in my plans during my undergraduate program, and even for the first few years afterwards. But I owe a debt to Dr. Jacek Ilow for setting me on the pathway to [eventually] pursuing one. It was under Dr. Ilow's guidance I had my first publication as part of my undergraduate degree, and with that an introduction to academic research. Dr. Ilow was also instrumental in my decision to apply for the NSERC funding program, which gave me some flexibility on research topics.

It was then under Dr. Zhizhang Chen I started the PhD, and under his guidance that I discovered the area of embedded security that this thesis pertains to. Had he not provided the freedom for me to explore these areas, I am doubtful I would have had such a fruitful PhD. I know few PhD students who have had the opportunity to so completely explore their research area while still retaining careful guidance. Dr. Chen has also been instrumental in expanding my PhD experience to see the larger academic life including assisting other students, teaching undergraduate courses, and updating labs/courses.

Just as it takes a village to raise a child, so does it to raise a PhD student. My supervisory committee also includes Dr. Srinivas Sampalli (in addition to Dr. Chen & Dr. Ilow), and I remain grateful for their guidance at many PhD 'life events' which have been critical to refining my thesis topics. This has often included a substantial time commitment to review papers, proposals, and presentations. The final version of the thesis here also includes comments and suggestions from both my committee and the external examiner, Dr. Jens-Peter Kaps. I'm grateful for their detailed remarks, and giving me a last chance to fix errors before the final version is committed in perpetuity.

Of course, I owe a great debt to my parents John & Eleanor for sending me on the path of electrical engineering. They encouraged me from a young age, even though they did not work in this area themselves. They would take me to electronics shops and stores where I would spend what I'm sure (in retrospect) was an unreasonable

amount of time looking at parts, skimming books, and asking questions of store owners. Nor did they question the solder marks on the kitchen table, magazine subscriptions, and electronics kits that didn't always work.

This thesis builds on the work of many that have come before me, and I'm grateful for both their open research, along with their willingness to answer questions from an unknown PhD student. This includes those working on open-source software and hardware, which this thesis makes use of. I hope I can contribute to future work by releasing much of the work I have completed in this thesis under open-source licenses.

During the completion of this thesis, the support of my family has been instrumental. My wife Hilary has put up with many late nights, my travel to conferences, and once again solder marks on the kitchen table. I remain eternally grateful for her love, support, and encouragement for me to switch to lead-free solder.

Chapter 1

Introduction

1.1 Background

The Internet of Things (IoT) is one of the most popular markets of electronic devices, with forcasts expecting over $10 trillion a year of economic value by 2025 [63]. While some would question if an internet connected pet feeder is truly needed (and the resulting problems that happens when it breaks down due to a server failure[83]), the IoT market is a demonstration of what is possible with continuing advancements in embedded computer systems.

Embedded computers are typically a small task-specific computer contained within a larger system. Embedded computers are found in almost every market: within consumer goods you would find one within a smart thermostat (such as a Nest), digital camera, or an alarm system; within the automotive environment of typical car you could find 30 or more small computers, controlling everything from the engine to raising and lowering windows[131]; and within aerospace these embedded computers could be found running the autopilot on passenger aircraft, within the entertainment system, and logging critical flight parameters to the black box.

These embedded computers have continuously gained more processing capability, while at the same time requiring less electrical power to operate, and with a lower cost. As an example in 1996 a typical embedded processor for a low-cost application might be the Microchip PIC16C711, which had 1000-unit pricing of $3.53 (approximately equivalent to $5.36 in 2016 dollars)[86]. The PIC16C711 had 1KByte of EPROM program memory and 68 bytes of RAM, 4 channel ADC, and operates at up to a 20 MHz clock.

In 2016, for $0.62[1] at 1000-unit pricing one can get a STM32F030F4P6 device which has 16Kbyte of FLASH memory, 4KByte of RAM, an 11 channel ADC, and operates at up to a 48 MHz clock. For $4.93 a ATSAM4S16CA-AU device has 1MByte

[1]Pricing based on Digi-Key prices in October 2016

of FLASH memory, 128KByte of RAM, operates at up to 120 MHz, and includes many peripherals such as ADCs and USB. Even more advanced devices integrate WiFi and other wireless communication interfaces, and are available at very low cost in higher quantity.

The growth of more advanced processors being available at lower cost is of great interest from a security perspective. Many consumer electronics now run small operating systems, even if the task being performed is fairly simple. The Philips Hue is a smart lighting system, yet its main bridge runs Linux and contains two microcontrollers [102]. Often this is done to simplify development – writing very small embedded code from scratch is time-intensive, and may be difficult to modify or update in the future. But if the product is developed on-top of an OS (such as Linux), the developers can take advantage of the underlying OS for features such as networking, routing, file systems, and web communication.

These more complex systems often have more attack vectors someone could potentially exploit. And if they are able to get access to a system, they may be able to perform more damaging attacks. For example a Nest wireless thermometer may have a wireless interface connected to smart lights, and a wireless interface connected to the home network. Could someone attacking the smart lights use the Nest thermometer as a bridge to attack the home network?

Such bridging has already been demonstrated. In 2015 researchers showed how they could hack a Jeep Cherokee remotely, sending messages on the local network of the car over a cellular connection [55, 87]. These local messages allowed almost complete control of car systems - they could perform everything from turning on windshield wipers to turning off the engine. It was possible to perform these attacks only after significant reverse engineering of the embedded computers, and finding ways to remotely reprogram several embedded computers, before it was finally possible to bridge messages from the external cellular modem onto the local automotive CAN network.

Security considerations are a required part of embedded design in this environment. This becomes especially important as more devices are equipped with networking and communication capabilities, giving them the ability to be used as part of a larger attack, even if the specific device in question does not perform important

duties.

Typical solutions that are applicable on PCs, such as using strong encryption, may be either more difficult or considerably less effective on embedded systems. For example differential power analysis (DPA) [72] can break a perfect implementation of AES, making it trivial to attack a device even though it uses encryption to protect the firmware.

While many of the lessons learned protecting PCs from virus and security threats are relevant, there is a rather large additional body of knowledge on security threats that are specific to the embedded environment. Designers of embedded systems must be aware of these threats in order to design secure embedded systems.

1.2 Motivation

Despite the importance of security on embedded systems, many products are released with insufficient protection against attackers. These security threats should be well-known by now – DPA as a method of breaking encryption has been publically disclosed for 18 years. Yet the continual use of unprotected encryption routines shows that engineers are either unaware of the threat or do not take it seriously [107].

Before starting this PhD, I had worked at Atmel on various projects related to IEEE 802.15.4 wireless protocols, including the ZigBee-IP Standard [4]. During this time I worked on both embedded system design and protocol specifications, and part of this work included discussions around security considerations. In these discussions the types of attacks discussed in my thesis were relatively unknown by most engineers (myself included at the time), and generally it was assumed they were not a practical threat. This was almost entirely due to my unfamiliarity with performing them myself – it's difficult to gauge the relevance of these attacks without practical experience.

While engineers may be familiar with the existence of these attacks, it is very rare to find a general embedded engineer who has actually performed them. Universities doing this research might expose their undergraduate engineers to the problem, but the majority of engineering students would have seen these attacks only in a theoretical lecture, if they were mentioned at all.

It was clear to me that not only is more research needed in pushing these attacks forward, but it's needed to make this research available for use by both academics

and engineers in industry. Bridging this gap is critical for ensuring products being developed in industry remain secure for their expected lifetime. What is needed is that any embedded designer is not only aware of the attacks, but has actually performed them and is able to understand how they apply to systems they are designing.

1.3 Contributions of the Thesis

My motivation of both performing fundamental new research, along with helping to make this work available for a wider audience has shaped the direction of this thesis. The main contributions are related to three areas: design of a large open-source platform for performing the attacks, new methods of performing fault attacks, and further examples of attacking devices used in the development of products.

1.3.1 Capture Tools

The most major contribution made in this thesis for embedded side-channel power analysis is the ChipWhisperer open-source platform, discussed in Chapter 4. This platform was released as an open-source project, and has already been used in numerous academic papers by other researchers [19, 23, 40, 42, 43, 49, 61, 76, 79, 96, 97, 113, 148]. This platform contains both the hardware designs for building a measurement platform, along with the software required to drive this hardware and perform many types of attacks on embedded systems.

I had many talks at conferences related to this platform, including talks at industry-focused conferences such as Black Hat, DEFCON, SEC-T, and RECON. In addition I ran a tutorial at CHES on the use of this ChipWhisperer platform for building a very low-cost lab setup, and later used the ChipWhisperer platform as part of a capture-the-flag (CTF) event at CHES 2016.

The capture tool work also built into development of novel techniques for preprocessing data, discussed in Chapter 5. This work demonstrates methods of performing 'clock recovery' both using a regular oscilloscope, and using my special ChipWhisperer platform. This clock recovery hardware opens up the ability to synchronize certain fault injection attacks to a device with an internal clock.

1.3.2 Fault Attacks

Fault attacks may be one of the most powerful methods of breaking embedded systems. My work concentrated on the development of a simple 'crowbar' method of inserting faults, and demonstrating its advantages over previous systems. In particular, this method is able to achieve extremely high temporal accuracy (i.e., targeting a specific instruction). This method can also be readily applied to real devices, for example I target two different off-the-shelf embedded Linux computers in addition to an 8-bit microcontroller and a custom FPGA board. This is discussed in Chapter 8.

1.3.3 Attack Examples

Finally, several examples of side-channel power analysis attacks are presented. The Atmel ATMega128RFA1 device was analyzed in Chapter 7. This chapter also demonstrates how to attack the device when it's running a standard IEEE 802.15.4 wireless stack. This requires development of several new techniques to work with the AES-CCM* mode encryption used in this standard. It also required development of specific measurement techniques for use with this device, which are widely applicable to other IEEE 802.15.4 SoC devices from other manufactures.

I've also brought an example attack against an AES-256 bootloader, specifically one that was part of an app-note published by a microcontroller manufacture. This demonstrates how side-channel power analysis can be used to trivially break software implementations where engineers may blindly follow the suggested best practices from such app-notes.

1.4 Outline of Thesis

The rest of this thesis is organized as follows – background on aspects of embedded hardware security covered in this thesis is presented in Chapter 2. This includes side-channel power analysis and glitching attacks. The power measurement considerations are presented in Chapter 3, which are then built upon to develop the open-source ChipWhisperer platform in Chapter 4. Using this platform on practical devices may require the clock recovery techniques discussed in Chapter 5, as they allow these techniques to be applied in situations where no external clock is available. The next two

chapters are practical attacks: first Chapter 6 breaks an AES-256 bootloader, and then Chapter 7 breaks an AES hardware accelerator along with providing considerations for breaking it during use with IEEE 802.15.4 wireless protocols. Chapter 8 details the novel power glitching method developed, and demonstrates this against several devices including a Raspberry Pi and a FPGA device. Finally Chapter 9 has some final remarks on this thesis, and what extensions can be made. Appendix A explicitly details my contributions made to the field during this thesis.

Chapter 2

Background and Related Work

The literature review that follows outlines the history of side-channel power analysis and other embedded hardware attacks. This has been split into two general sections – side-channel power analysis, and fault injection. Certain topics which are outside the scope of this thesis are not discussed, such as attacks involving the decapsulation of chips and directly probing the surface or attacking the decapsulated surface [5, 124].

2.1 Side-Channel Power Analysis

It had been known that the power consumed by a digital device varies depending on the operations performed since at least 1998, when Kocher, Jaffe, and Jun showed the use of the power analysis for breaking cryptography[72]. The first example given was that of Simple Power Analysis (SPA), where knowing the sequence of operations would directly allow read-out of the secret key. Differences in power consumption for different operations allows breaking of cryptographic algorithms using SPA.

As an example, consider the source code from Listing 2.1. This code is taken from the file `bigint.c` of `avr-crypto-lib`, an open-source library for the AVR microcontroller. This particular function is used as part of the RSA crypto system.

When a bit of the `exp` variable is 1 a square and multiply is performed, and when a bit of the variable is 0 only a square is performed. Looking at the power consumption, we can see some difference between a square and multiply operations. This is shown in Figure 2.1, where the code has been compiled onto an Atmel XMEGA microcontroller. The leakage in Figure 2.1 can be seen in the timing when a '1' is processed compared to a '0'. While both the square and multiply have similar power signatures on this platform, the delay on entering the square routine is slightly longer. The delay marked at "A" in this figure is about 80 mS, and the delay at "B" is about 60 mS. The slightly longer delay can very reliably be detected to determine if two function calls have occurred (square + multiply, indicating a '1') or only one function

Listing 2.1: The following lines are from `bigint.c` in `avr-crypto-lib`, showing an example implementation of the vulnerable RSA code.

```
uint8_t flag = 0;
t=exp->wordv[exp->length_W - 1];
for(i = exp->length_W; i > 0; --i){
  t = exp->wordv[i - 1];
  for(j = BIGINT_WORD_SIZE; j > 0; --j){
    if(!flag){
      if(t & (1 << (BIGINT_WORD_SIZE - 1))){
        flag = 1;
          }
    }
    if(flag){
      bigint_square(&res, &res);
      bigint_reduce(&res, r);
      if(t & (1 << (BIGINT_WORD_SIZE - 1))){
        bigint_mul_u(&res, &res, &base);
        bigint_reduce(&res, r);
      }
    }
    t <<= 1;
  }
}
```

call (square, indicating a '0').

This particular variable that is leaked in this manner is not an arbitrary one, but instead knowledge of this variable leaks the value of the secret key used in this operation. Thus SPA allows us to directly break the secret key used during the operation.

While SPA is capable of breaking cryptography by deciphering operations, the same paper also presented a more powerful attack called differential power analysis (DPA) [72]. This seminal work demonstrated that there may be considerable problems with implementations of otherwise secure protocols on embedded hardware devices. In particular, this introduced the idea that measurements of the power could actually reveal something about the *data* on an internal bus, and not simply the

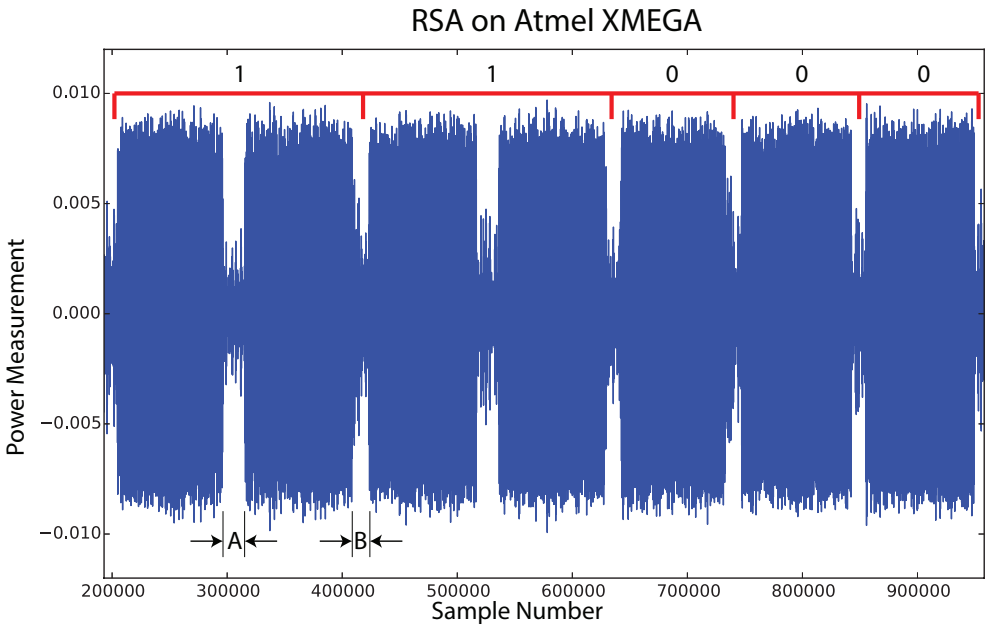

Figure 2.1: This exploits the data-path dependent code from Listing 2.1, which allows us to read the secret data off bit-by-bit.

overall operation.

Fundamentally, this is due to physical effects of how digital devices are built. A data bus on a digital device is driven high or low to transmit signals between nodes. The bus line can be modeled as a capacitor, and we can see that changing the voltage (state) of a digital bus line takes some physical amount of energy, as it effectively involves changing the charge on a capacitor.

There are several ways to use this knowledge. I'll discuss the main types of attacks used within this thesis, as well as summarizing some of the more recent work in the following sections.

2.1.1 Attack Types

The initial attack presented in [72] caused a digital device to execute an operation with both known and secret data. If we consider the case where that known and secret data is mixed together, we could define the known data as P, the secret data as K, and the operation as $C = f(P, K)$.

The DPA attack measures power consumption of the device during this operation.

We can measure $i = 0, \cdots, N$ such operations with random known input data P_i, and constant unknown secret data K. We could set K to some assumed value K'. Assuming that K is a single byte, this presents 256 possibilities for the value of K'.

For each possibility of K', we could have a group of hypothetical outputs of the operation $C'_i = f(P_i, K')$ for each known input P_i, again where $i = 0, \cdots, N$. At this point we wish to determine which value of K' matches the true value of K on the hardware device running the algorithm.

One method presented in [72] is to target a single bit of the value of C' (and hence K'). For each hypothetical value of K' we can separate the power traces into two groups: one where a bit of C'_i is '1', one where the bit is '0'. If our hypothetical value of K' matched the true value K, we would expect a difference at some point in the mean power consumption between the two groups.

If our value was incorrect, we would expect no such difference, as the grouping could simply be considered as a random grouping of the traces into the two sets. In practice, such difference does exist when correctly grouped. Fig. 2.2 shows an example of the difference between the mean of two such groups, which have been correctly grouped into a set where the internal bit is '1' and the set where the internal bit is '0'. Note the trace shows us the location in time where the data is manipulated, as all other samples where the processor is *not* handling the data we targeted have the same mean.

This demonstrates that on a fundamental level devices do leak information regarding the state of the internal data bus. One additional consideration is how this can specifically be used to break cryptographic implementations, as it would appear this method still requires some level of "guess and check". This "guess and check" however is not performed over the entire key-space.

The implementation of cryptographic algorithms involves operations on individual bytes or words of data. For example although AES-256 involves a key of 256 bits (32 bytes), the "guess and check" for performing DPA only involves guessing a single bit at a time. This means a very tractable problem of performing $2^1 \times 256$ guesses, something even a typical personal computer can accomplish in a few seconds.

The hypothetical output of some function we are targetting is typically referred to as the "intermediate value", as we are targetting some value within the entire

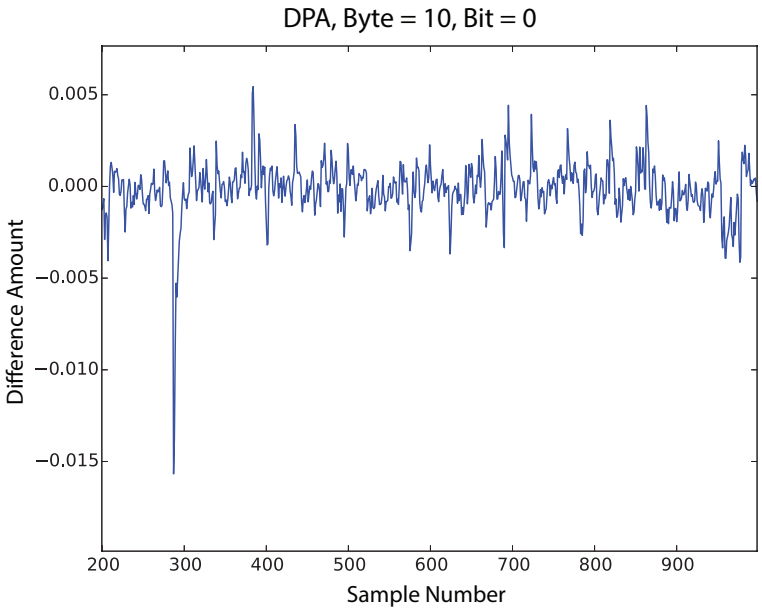

Figure 2.2: This demonstrates a DPA attack on a single bit, the large spike occurs at the instance in time where the processor is manipulating the data of interest.

operation of the algorithm. When attacking AES this is often after the first round of the SubBytes operation, as the non-linear property of the SubBytes improves our attack by eliminating the linear relationship between the input and intermediate values. In addition one byte of the plaintext will directly mix with one byte of the secret key, reducing the complexity of performing the guess and check operation.

While the DPA attack was the first proposed methods, more efficient methods of discovering the secret key information using the power traces exist. We'll discuss two major methods next: the Correlation Power Analysis (CPA) attack and template attacks.

Correlation Power Analysis (CPA)

Whereas DPA looked at simple differences between two groups of data, the CPA attack develops more precise assumptions on the power consumption and relation on an internal data bus. The CPA attack was first present in 2004 by Brier et al. in [22], and will be summarised here.

For a simple 8-bit microcontroller, we can use a "leakage model" that suggests the instantaneous power consumption is related to the number of bits set to '1' on

the internal databus. This assumption is based on two factors: (1) our previous knowledge that moving the state of a line takes a certain amount of power, and (2) knowledge that microcontrollers set their internal buses to a constant state before the final value is loaded.

This constant state is known as the 'precharge' state. This precharge has been used since the early design of microcontrollers, where it was easier and faster to design a bus with precharge logic to pull the bus to the '1' state, requiring each module driving the bus to only have the pull-down logic (rather than requiring full push-pull and enable transistors on each bus connection)[84].

More recent devices may pre-charge to other levels, such as precharging to a level between '1' and '0', with the objective being to reduce the power and time required to transition to the final level [53]. This pre-charge would require push-pull drivers at each bus connection, so is targeting improved performance rather than a simplified design.

From an attack perspective, specifics of the pre-charge are irrelevant. Instead the attacker cares there is a constant starting level, meaning a linear relationship between the number of bits set to '1' on the databus and the power consumption. Depending on the precharge level and measurement style this relationship may have a positive or negative slope. Without this pre-charge we instead have a relationship between the *change* in bits between two bus states, and thus would also need to know (or guess) the previous state.

The case of the pre-charge will be referred to as the Hamming Weight (HW) model, where leakage is assumed to be related to the number of bits set to '1' on the bus. Without the precharge we would have the Hamming Distance (HD) model, where the leakage is related to the number of bits changing states on the bus.

As a validation of this previous work, I have measured the power consumption of an 8-bit microcontroller (Atmel ATMega328P) at the moment it is manipulating data with various number of bits set to '1'. The results in Fig. 2.3 show an excellent relationship between the HW of the data and the power measurement.

The basic equation for a CPA attack, where $r_{i,j}$ is the correlation coefficient at point j for hypothesis i, the actual power measurement is $\mathbf{t_{d,j}}$ of trace number d at point j, and $p_{d,i}$ is the hypothetical power consumption of hypothesis i for trace

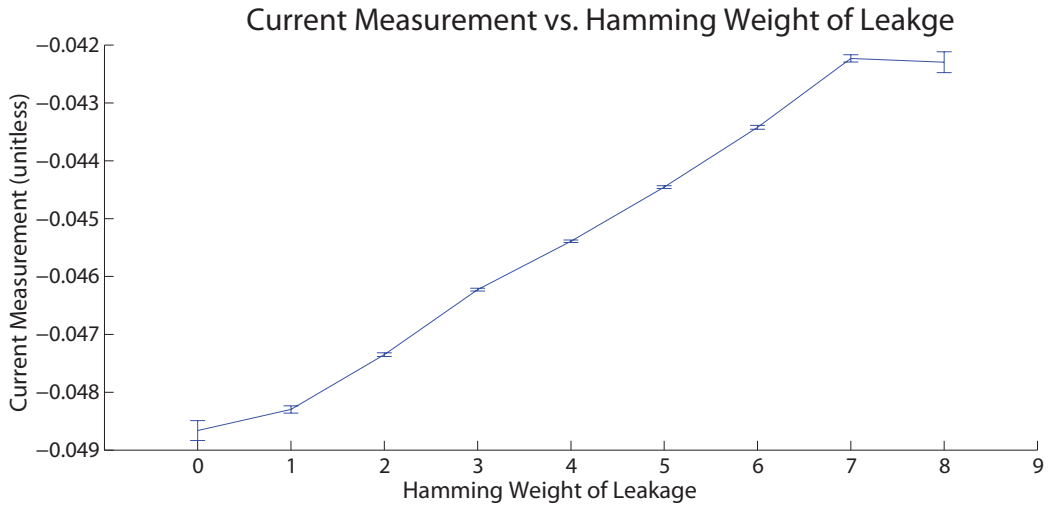

Figure 2.3: Power consumption of device under attack performing an operation on data with different Hamming Weights (HW), showing the average current consumption of the AtMega328P microcontroller for each possible hamming weight of an 8-bit number. Error bars show 95% confidence on average (based on the sample standard deviation).

number d, with a total of D traces is given in equation (2.1). This equation is simply an application of the Pearson's correlation coefficient given in equation (2.2), where $X = \mathbf{p}$, and $Y = \mathbf{t}$.

$$r_{i,j} = \frac{\sum_{d=1}^{D} \left[\left(p_{d,i} - \overline{P_j} \right) \left(t_{d,j} - \overline{t_j} \right) \right]}{\sqrt{\sum_{d=1}^{D} \left(p_{d,i} - \overline{P_j} \right)^2 \sum_{d=1}^{D} \left(t_{d,j} - \overline{t_j} \right)^2}} \tag{2.1}$$

$$\rho_{X,Y} = \frac{\operatorname{cov}(X,Y)}{\sigma_X \sigma_Y} = \frac{E\left[(X - \mu_X)(Y - \mu_Y) \right]}{\sqrt{E\left[(X - \mu_X)^2 \right]} \sqrt{E\left[(Y - \mu_Y)^2 \right]}} \tag{2.2}$$

The form given in these equations is referred to as the *normalized cross-correlation*, and frequently used in image processing applications for matching known templates to an image.

CPA as Matched Filter

While the CPA attack has been developed independently of similar work in communications theory, it is useful to recognize the parallel between the CPA attack and basic methods of recovering a signal in communications theory.

In communications theory, the most basic problem statement is how to receive a signal that has been corrupted by Additive White Gaussian Noise (AWGN). The continuous-time and discrete-time interpretations of this problem are given as follows:

$$r(t) = s(t) + n(t) \tag{2.3}$$

$$r[n] = s[n] + w[n] \tag{2.4}$$

The transmitted signal or sequence $s(t)$ or $s[n]$ is one of several valid signals, the specific signal depending on the system. The objective of the communication systems is for the receiver to determine which of the possible symbols $s_1(t), s_2(t) \cdots, s_N(t)$ was sent based on the received signal $r(t)$.

The objective of receiving a known signal in Additive White Gaussian Noise (AWGN) has a well known solution, the matched filter (or 'North filter'), first described in 1943[99]. For a known signal $s(t)$ transmitted over a channel with AWGN $n(t)$, at the receiver we would have the received signal $r(t)$:

$$r(t) = s(t) + n(t)$$

Which is then passed through the matched filter with impulse response $h(t)$:

$$y(t) = r(t) * h(t) = \int r(\tau)h(t-\tau)d\tau$$

If we sample the output $y(t)$ at $t = T$ to make our decision, we will use the matched filter with the following impulse response:

$$h(t) = s(T-t), 0 \leq t \leq T$$

Thus leading us to find that:

$$y(t) = \int_0^t r(\tau)s(T-t+\tau)d\tau$$

Finally the value when we sample at T will become:

$$y(T) = \int_0^T r(\tau)s(\tau)d\tau$$

It can be noted this is equivalent (for $t = T$) to the cross-correlation of $r(t)$ and $s(t)$ at time $t = 0$:

$$y(t) = r(t) \star s(t) = \int r(\tau)s(t+\tau)d\tau$$

$$y(0) = \int_0^T r(\tau)s(\tau)d\tau \tag{2.5}$$

The use of cross correlation for matching a known template is well known in computer vision as well, for example the use of cross correlation is presented in [77] with approximately the same form as equations (2.1).

The forms given in equations (2.1) and (2.2) force both $r(t)$ and $s(t)$ to be zero-mean and normalized by standard deviation. This is necessary as we do not have proper scaling of the template $s(t)$ used at the receiver.

Note again $r[t]$ is the received signal, and $s[t]$ is the transmitted signal. One critical difference between communications systems and side-channel power analysis is the definition of the argument of $s(t)$. In communications we are sending a known signal $s_n(t)$, which may be drawn from a set of 'allowed' signals in the set $s_1(t), s_2(t), \cdots, s_N(t)$. Each of these signals is typically a finite-length signal as a function of time (or samples in the discrete case). At the receiver we can use the matched filter to determine which of the N possible signals was transmitted.

For side-channel analysis, our function $s(t)$ is actually defined over the number of cryptographic operations we observed. In equation (2.1) this was d, the 'trace index', and thus will be referred to as $s(d)$. Each of the possible functions $s_1(d), s_2(d), \cdots, s_N(d)$ reflects the hypothetical value for the byte of the secret key we are attacking. Thus the matched filter comparison is always done at the same sample (i.e. time point) in each power measurement trace $\mathbf{t_d}$.

Template Analysis

The previous descriptions used an intuitive description of the physical hardware to produce a leakage model. For example we assumed a correlation between the number of ones on a databus and the power measurement at a single point in time. We can perform much more advanced analysis if we assume we have a copy of the device

under our control; for example when attacking a specific IC with a secret key, we may have another IC from the same batch which we can program with a known secret key.

In this manner we will not make assumptions about the leakage. Instead we will build 'templates' of known leakages for specific values or attributes of the secret key using our copy of the device. We can then compare leakage from the actual device with our templates to determine the most likely value of the secret.

This template attack makes it possible to recover a secret key using only a single measurement, as we can build templates for each value of each byte of the secret key. Template attacks were published before the CPA attack, being described in 2002 by Chari et. al. [30].

To describe the template attack, I'll first take the case of the CPA attack where we consider only a single point of interest at a time. To generate a template set we could keep the input plaintext P_d fixed for each power measurement t_d. We'll instead vary the known encryption key K, such that each trace has a known and random key K_d, again where d is the trace index. Assuming the key K_d is made of a number of bytes (such as AES-128 having 16 bytes for each key), we can further look at each byte j of the key used for trace d, called $K_{d,j}$.

With a set of power traces t_d there is thus an associated value for the secret key K_d used on each trace d. Looking at the entire set of power traces, the power traces could be then grouped into different groups $\mathbf{t_{K_j=g}}$, where g is the value of secret key byte K_j in the range of $\{0, 1, \cdots, G\}$. For example if we were attacking AES-128, we would first look at byte $j = 0$. Because this is a single byte, we would make 256 sets of power traces, one for each possible value of $K_0 = \{0, 1, 2, \cdots 255\}$. If $D = 100000$, there would be about 390 power traces in each set.

Assuming we have a single point of interest $l = l'$ within each trace $t_{d,l}$, we could determine the sample mean $\overline{\mu}_g$ and variance $\overline{\sigma^2}_g$ of each population. Being given a trace where the secret key is *unknown*, the problem becomes one of simply determining the most likely population the associated power trace for the unknown secret key belongs too. This gives us the value for g, which can then be mapped to the value of a secret key byte.

In practice, the template attack is extended to represent multi-variate random variables. That is rather than taking a single point in the template, we consider

multiple points within the power trace $t_{d,l}$. This improves the templates ability to distinguish between the various populations in selecting the most likely value of the unknown byte.

Note templates do not necessarily need to directly provide the exact value of the unknown secret key. Instead a template could be built for possible Hamming weights of some known intermediate value, and the result of the template is used in a similar manner to the CPA attack to determine the most likely secret key value.

One of the difficulties of applying a template in real-life applications is finding a closely matched "copy" of the device under test. In an ideal situation, the attacker would be able to reprogram the actual device being attacked. Practical work has demonstrated that template attacks can still be applied even when this is not possible, although additional work may be required for a successful attack [109, 31].

While template attacks are not used in my thesis results, I have built several examples of their use and application available on the ChipWhisperer wiki at `http://www.chipwhisperer.com`. More details of the ChipWhisperer project will be presented in later chapters.

2.1.2 Implementation Leakage and Success Rate

Measuring the effectiveness of an attack toolchain is critical to understand and compare attack methods, capture hardware, and countermeasure potency.

There are two general methods of performing this assessment. The first method is to perform an empirical measurement of how successful a given attack was against a specific measurement setup. This success-based measurement demonstrates how well an entire attack works, and the specific amount of work required to break a given implementation. This measurement can be affected by a change anywhere in the attack toolchain – from physical setup differences to attack algorithm. A number of such measurement techniques are detailed in [128], and for example used in the DPA Contest v2 results summary [34].

The second general method is to measure an indicator of the "leakage" coming from a given system. This simplifies analysis of systems where the most effective exploitation technique of a specific system is unknown. It is possible for example a system is highly secure against one specific attack, but even a slight change in attack

setup will cause the system to be highly vulnerable.

The leakage-based measurement does not demonstrate *how* to break a system, and often provides only a lower bound on the work required to break the system. It is possible a system may be considered highly leaky based on these metrics, but a practical method of attacking the system is unknown.

Global Success Rate (GSR)

The Global Success Rate (GSR) is an empirical performance measurement that directly measures if an attack was successful. This measurement can directly demonstrate how difficult it is to completely break an implementation using side-channel power analysis.

If the attack algorithm has access to N traces, we can consider the attack successful if the algorithm successfully determines the correct encryption key with N traces. We can present a number of different sets of N traces, and average the number of times the entire encryption key was successfully recovered with a set of size N.

This gives us the 'global success rate', where a rate of 1.0 means the attack always succeeds. Typically we will consider an attack successful for a GSR about 0.8, i.e. given a specific number of traces, the attack succeeds 80% of the time.

The GSR only indicates when the attack is completely successful – in reality it is sufficient to reduce the guessing entropy to a tractable level, instead of requiring the attack to directly give us the complete encryption key. Another metric which provides a measure of the reduction of guessing space is discussed next.

Partial Guessing Entropy (PGE)

The 'guessing entropy' is defined as the "average number of successive guesses required with an optimum strategy to determine the true value of a random variable X"[81]. The 'optimum strategy' here is to rank the possible values of the subkey from most to least likely based on the value of the correlation attack (higher correlation output is more likely).

The 'partial' refers to the fact that we are finding the guessing entropy on each subkey. This gives us a PGE for each of the 16 subkeys. A PGE of 0 indicates the subkey is perfectly known, a PGE of 10 indicates that 10 guesses were incorrectly

ranked higher than the correct guess.

The attack algorithm is given access to $1, 2, \cdots, N$ traces, and the PGE for each subkey is calculated. To improve consistency the PGE for each subkey is averaged over several attacks (trials). Finally, we can average the PGE over all 16 subkeys to generate a single 'average PGE' for the attack.

Leakage Measurement

The PGE and GSR require an attacker to know *how* to break a device. That is knowledge of the algorithm under test, leakage type, and particulars unique to the hardware or software cryptographic functions. Often designers of secure hardware only care about the *existence* of leakage. This means not necessarily determining how to break a device, just if a device appears to have some leakage that might be exploitable.

One of the most popular techniques uses a test-vector leakage assessment (TVLA), detailed in [54, 35]. This applies a set of preselected test vectors (test vectors being the input and secret key), and performs a statistical test on the power measurement during the test vector application. Briefly, these tests are typically applying a test vector from one of two 'groups' (such as a random or fixed group). If it is possible to determine a difference between the groups using side-channel measurements, this suggests there must be some information leakage.

The TVLA is typically performed by applying Welch's t-test to the two populations, where the two populations being tested are the power measurements recorded during each of the two different test vector groups. The t-test allows us to determine if the two measurement groups are distinguishable or not – we can simply specify some desired confidence, and the t-test provides a simple numerical output which is either above or below this threshold.

Assume the two groups were formed by grouping the complete set of traces $\mathbf{t}$ into $\mathbf{t} \in g_1$ and $\mathbf{t} \in g_2$. The set mean, variance, and cardinality is denoted by $\mu(\mathbf{t} \in g_n)$, $\sigma^2(\mathbf{t} \in g_n)$, and $|\mathbf{t} \in g_n|$ respectively.

Performing Welch's t-test requires the application of 2.6 to the two data sets. This test is used to determine the validity of the null hypothesis that the two sets of samples come from the same population. If the test suggests we cannot reject the

null hypothesis, this means we cannot determine which population g_1 or g_2 a given sample came from using the power measurement.

$$t = \frac{\mu\left(\mathbf{t} \in g_1\right) - \mu\left(\mathbf{t} \in g_2\right)}{\sqrt{\frac{\sigma^2\left(\mathbf{t} \in g_1\right)}{|\mathbf{t} \in g_1|} + \frac{\sigma^2\left(\mathbf{t} \in g_2\right)}{|\mathbf{t} \in g_2|}}} \tag{2.6}$$

Typically, the t-test is used with a fixed threshold where if $|t| > 4.5$, it indicates an ability to determine which population g_1 or g_2 a given power trace measurement came from. This means the device has some data-dependent leakage we may be able to exploit. See [54] for details of this threshold, including determining specific values for desired confidence intervals and with certain set sizes.

Care must be taken in applying the t-test, as it simply indicates a difference in the mean of the sample groups, and not specific cryptographic leakage. All other (non-cryptographic) leakage must be carefully controlled – for example one cannot apply all of the test vectors from group 1, and then all of the test vectors from group 2. There will likely be a difference in the means between these two groups due to variation in environmental conditions (such as voltage and temperature), which the t-test metric will detect. Full consideration of this is discussed in [54, 35].

An improvement to the basic t-test is presented in [41], which also includes a reference to a number of academic publications showing the use of t-test results in analyzing hardware devices. In particular [91] shows an example of the use of a t-test to determine an appropriate leakage model, and then applying this leakage model with CPA to break a hardware AES implementation.

2.2 Fault Injection

Faults cause a computer program to behave in an unintended manner. For many systems this could have dire consequences, and protecting systems from such faults is an important area of research. Fault injection encompasses the techniques which are used to purposely cause faults to occur, for example as part of validating or testing a fault-tolerant or fault-detecting scheme[116]. Fault injection is also useful as a testing tool when designing for environments likely to cause single-bit failures, such as space applications or high-radiation environments [25].

Fault injection is also a powerful tool to break cryptographic algorithms. The previous example assumed the 'dire consequences' of a fault occurring were a result of the system performing an unexpected action. But a fault could be purposely injected to cause a system to behave abnormally, to an attackers advantage. It has been well known that a variety of fault injection methods can be used for this purpose[6, 14].

Previous work on fault injection has demonstrated methods of breaking cryptographic algorithms such as DES[20], AES[45, 32, 16], and RSA[21, 120, 15] by introduction of faults at specific parts of the algorithm. Of these, a practical demonstration of the proposed method is also given in [32, 120, 16, 15]. All of these demonstrations are performed on a custom board, specifically designed to inject faults into the embedded computer running the cryptographic algorithm. The reader is referred to [17] for a more detailed survey of attacks on AES and RSA.

2.2.1 Non-Cryptographic Attacks

Fault injection is powerful as it can be used to attack almost any aspect of an embedded system (not just cryptographic functions), but still achieve very damaging results. For example consider the C code of Listing 2.2, which is a hypothetical implementation of a response buffer in an embedded system. A fault attack could cause the value of i (or *strlen*) to become corrupt. This is made worse as compilers may replace the 'less than' comparison with a simple 'branch if not equal', which is only looking for the exit value. A fault which causes the i value to become larger than strlen will print the entire memory after the *str* buffer until the counter rolls over.

If this was a 32-bit system, the default integer size (of 32-bits) would mean it could dump the entire memory address space. If both RAM and code memory exist on the same address space (common on processors such as ARM), this would allow an attacker to receive both the code and data memory contents. This can be applied on more complex systems, such as M. Scott demonstrated by glitching a device with USB to dump an entire firmware image [121]. The work in [121] used the ChipWhisperer platform to inject a voltage glitch for this attack.

Listing 2.2: A simple `puts()` style routine with an added length field to avoid issues with unterminated strings.

```
void puts(char * str, int strlen){
  for(int i = 0; i < strlen; i++)
    putch(*(str + i));
  }
}
```

2.2.2 Methods of injecting faults

A detailed survey of fault injection methods will be presented in Chapter 8, so only a brief summary will be replicated here.

The two earliest methods discussed are tampering with the clock and voltage of the target device, which were first published extensively about in 1996 [5]. Clock glitching was seen as a reliable method of introducing faults at specific clock cycles, as it introduces some perturbation to the clock that causes setup and/or hold times of internal registers to be violated. This causes incorrect data to be loaded into internal blocks, which can cause effects such as the wrong instruction to be executed, incorrect data used in a calculation, or incorrect program counter changes.

More advanced methods include targeting specific areas of the chip surface. This can be done with various forms of electromagnetic radiation from EM pulses at the lower frequency range to optical pulses at the higher frequency range [60, 95, 124]. This thesis focuses on voltage and clock glitching only, as discussed in Chapter 8 (with some additional details in Chapter 4 and Chapter 5).

Chapter 3

Power Measurements

This chapter is based on previously published material from my paper at [101].

In order to perform side-channel power analysis, we must have a method of measuring the power consumption of a device under test. There are two general classes of probes used for measuring power consumption: a resistive shunt as used in the seminal DPA work [72], or an electromagnetic (EM) probe [50]. EM probes have been shown to result in more successful attacks [129], with the advantage that EM probes do not require any modification to the device under attack, and can even be performed at a moderate range [65]. The specifics of attacking a given device (and the possibility to perform this at a range) will vary greatly with specifics of the leakage, the device being attacked, and the equipment available to the attacker.

Many types of EM probes have been used in published work, including commercially manufactured probes. Comparison of different probe constructions is found in [39, 82]. Smaller probes can be scanned over the chip surface to pick out specific features, such as bus/data lines [132].

By examining the underlying physical characteristics of the embedded platform, we can simplify the measurement requirements by both identifying ideal locations for placement of the probe, and improving the digitizer that generates digital samples of the measurements. This work will be used to develop the side-channel power analysis platform in Chapter 4, which is used in the rest of this thesis.

3.1 Emissions from Decoupling Capacitors

A decoupling capacitor is designed to provide a low-impedance path for high frequency current, as typically drawn at the clock edge[126]. For side-channel analysis with a resistive shunt, the decoupling capacitor significantly worsens the measured signal [69]. The higher-frequency components, which are of interest for SCA, are flowing through the decoupling capacitor and not the shunt.

Measuring the current through a decoupling capacitor for side-channel analysis was first explored in [37], which used a current transformer to measure the current flowing through individual decoupling capacitors. Current transformers use the principle of induction, which dates back to Faraday's discovery in 1831 [47], to measure current flowing in a conductor without the necessity of breaking the conductor. Using induction to measure current through a decoupling capacitor in-place has also been demonstrated, but such papers employed the measurements for the design of power distribution systems, and not for side-channel analysis [143, 144, 78]. This chapter builds on such previous work by looking at the performance of the inductive pickup for side-channel attacks, and the physical considerations for its use.

The method thus proposed is to wrap the target decoupling capacitor in a thin magnet wire, and connect this to the acquisition oscilloscope. Physically, this proposed method requires no modifications to the device under test. The localized nature of the measurement provides excellent rejection of interference, and the performance when used in side-channel attacks will be demonstrated to be slightly superior to other common methods.

3.1.1 SASEBO-GII CPA Measurement Setup

The Side-channel Attack Standard Evaluation Board (SASEBO) version GII from the National Institute of Advanced Industrial Science and Technology (AIST) in Japan provides a useful reference platform for performing side-channel analysis attacks. Characterizations are available in literature of the performance of this board under various attacks [69, 89]. The attack used here is a simple Correlation Power Attack, for which the reference code is available from AIST [119], with the cryptographic core under attack being the AES core provided for the DPA Contest Version 3 [119].

The performance analysis here consists of the number of traces required for the global success rate (GSR) to stay above 80%. This performance analysis was chosen to match recent publications of a similar nature [44, 128].

The measurement equipment consists of an Agilent 54831B Infiniium DSO as a reference, and the OpenADC platform (to be presented in 3.3) as a demonstration of low-cost capture hardware. The acquisition from the Agilent 54831B is done with

code from AIS T[119] which has been modified to support the scope being used, with a sampling rate at 2 GS/s. This scope does not support an external clock input. Vertical voltage scale differs depending on the measurement setup being used. For the OpenADC the sampling clock (96 MHz) is 4x the AES Core Clock (24 MHz), which is derived from the actual AES Core Clock. The OpenADC capture software is written in Python, and I have released the source code at `http://www.assembla.com/spaces/openadc`.

In all cases the internal voltage (VINT) of the FPGA is adjusted to 1.000 volts; this avoids any unintentional results occurring because the insertion of the current shunt will naturally reduce the voltage seen by the FPGA. The SASEBO-GII is equipped with a small adjustment range on the VINT voltage to null out the current shunt loss.

The board as shipped did not have decoupling capacitors mounted on VINT, which correspond to C46 - C52. Where a decoupling capacitor is mounted in these tests, only a single 100 nF capacitor is mounted on C46, for which a Murata part number GRM155R61A104KA01D size 0402 capacitor is used.

Current Shunt

The SASEBO-GII board provides connections for measuring current used by the cryptographic FPGA via a 1-ohm current shunt. This measurement uses the 'VINT' supply for the FPGA, which is measured at J2. This measurement is performed both with C46 mounted and unmounted.

H-Field Probe

An H-Field probe was constructed from a loop of semi-rigid coax. When using the 54831B oscilloscope, a MiniCircuits ZFL-1000LN Low Noise Amplifier (LNA) boosts the signal to achieve a better response. The OpenADC is directly connected to the H-Field probe, as the OpenADC contains an integrated LNA. A photo of the magnetic field probe is shown in Fig. 3.1. Detailed information about the construction process is found in [126], with some additional examples for side-channel analysis in [39].

Figure 3.1: Shielded magnetic field probe, before wrapping in an insulator to allow safe probing of any area of the device under test.

Shunt Measurement on Individual Capacitors

The current through an individual capacitor was measured with a 0.22 ohm current shunt placed in series with the capacitor. The voltage was read directly from the current shunt and fed into the oscilloscope.

Power Pin Measurements

If the decoupling capacitors are not mounted, the device will naturally see drops in its voltage supply as measured at the power pin, since the power distribution system is unable to provide a low-impedance source close to the power pin. For the SASEBO-GII board, the measurement is taken on the underside of the board, on the positive pad of the decoupling capacitor specified. Each decoupling capacitor pad aligns directly with the power pin of the cryptographic FPGA, see Fig. 3.2

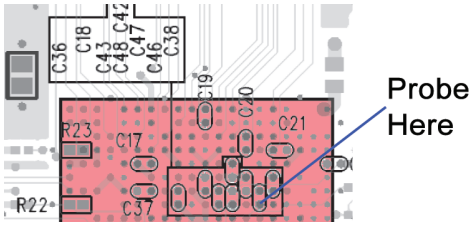

Figure 3.2: The decoupling capacitors line up directly with the power pins; if the capacitors are not mounted this provides a good source to measure the ripple on the voltage rail due to high-frequency power demands. The pink square is the location of the chip under attack on the top side of the board.

Inductive Wrapping

The proposed inductive wrap method uses 7 wraps of AWG34 magnet wire around the decoupling capacitor C46. One end of the magnet wire is soldered to the negative pad of the capacitor. The other side of the wire connects through a low-noise amplifier (ZFL-1000LN) for the DSO, or directly to the OpenADC. Fig. 3.3 shows a detailed photo of this setup.

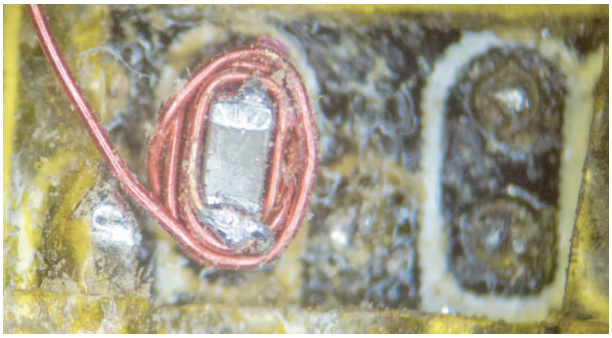

Figure 3.3: 7 wraps of AWG34 magnet wire around a 0402 capacitor. The yellow visible around the capacitor is Kapton tape used to isolate the rest of the PCB.

3.1.2 Measurement Results

Results for the Global Success Rate (GSR) of the CPA attack are shown in Fig. 3.4; Table 2 provides the number of traces require for the GSR to exceed 0.8. All of these measurements are taken with the Agilent DSO, a comparison between the DSO and OpenADC platform is given in Fig. 3.9.

Table 3.1: Traces required to achieve 1st order Global Success Rate (GSR) higher than 80% with a Correlation Power Analaysis (CPA) attack for several measurement techniques.

Measurement Method	Traces for GSR > 0.8
VCC-INT Shunt Measurement	4800
VCC-INT Shunt Measurement w/ decoupling	>5000
Inductive Pickup w/ decoupling w/ amplifier	3450
H-Field Probe w/ decoupling w/ amplifier	3850
Decoupling capacitor shunt w/ decoupling	4350
Voltage Probe	4550

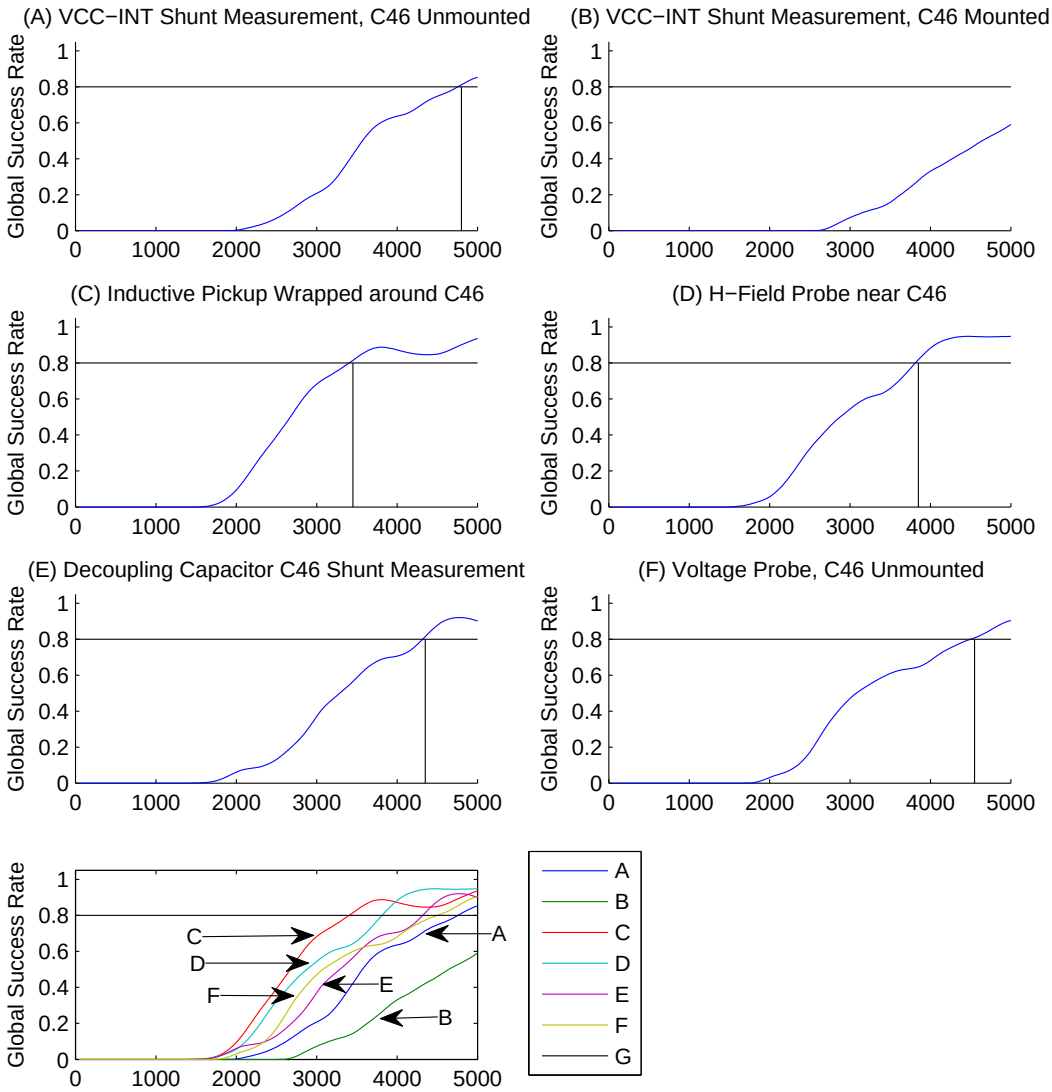

Figure 3.4: The first order global success rate (GSR) vs. the number of traces processed for a simple CPA attack. *A* through *F* show different measurement techniques; the final figure shows a comparison of the first-order GSR for each of the measurement techniques. The vertical lines show the intercept of the 1st order GSR exceeding 0.8, where the numeric value of these intercepts is given in Table 3.1.

Current Shunt, H-Field Probe

In order to confirm the test setup, several of the results duplicated work done elsewhere. For example, the shunt measurement on the entire VCC-INT power system

was expected to perform poorly when the single decoupling capacitor was mounted. This can be seen by comparing Fig. 3.1-A to Fig. 3.1-B. In addition, the H-Field probe should provide better results than the shunt measurement in order to agree with [129]. This is confirmed by looking at Fig. 3.1-D.

Inductive Wrapping

It can be see that the proposed measurement technique requires the smallest number of traces to achieve a GSR higher than 80% (>0.8). The signal from this technique is considerably stronger than with the H-field probe. The measured signal from the inductive wrap technique is about 10x larger in amplitude (Vp-p) than that from the H-field probe.

The stronger signal slightly relaxes the requirements of the amplifier, and means that the resulting SNR will be better compared to the H-field probe. The results here show slightly better performance for the inductive wrapping technique compared to the H-Field probe due to this improved SNR. The number of wraps used does appear to impact the GSR, as shown in Fig. 3.5. Here 7 wraps results in a better GSR than 2 wraps - the 7 wraps again resulted in a stronger signal, reducing noise in the measurement front-end.

Shunt on Decoupling Capacitor

The results here confirm the decoupling capacitor measurement does provide an improvement over attempting to measure the current drawn through the entire system. The performance is still lower than electromagnetic techniques; it is assumed that adding the shunt reduces the impedance of the capacitor, thus reducing the current which flows through it. In [37] a Current Transformer (CT) is used instead of a resistive shunt. Inserting the CT would also slightly increase the impedance, since the CT must be clamped around a wire in series with the decoupling capacitor.

Voltage Probe

The voltage probe is an extremely simple method of measuring local variations in the current demand. It does require the decoupling capacitor to be removed: for the best signal it would likely demand all nearby capacitors to be removed, as the nearby

capacitors provide some additional decoupling that dampens the signal. For devices under attack which require the decoupling capacitors to run, this method may not be possible.

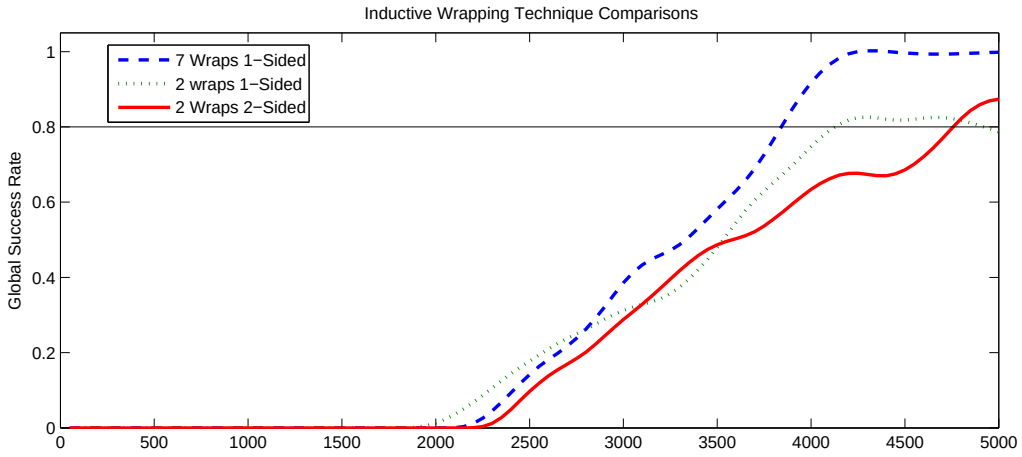

Figure 3.5: Comparison of different variations of the inductive wrapping technique. The maximum number of wraps was set based on the physical ability to keep the wraps around the decoupling capacitor. An '1-sided' wrap has one end soldered to the ground pad of the capacitor as in Fig. 3.3, where a '2-sided' wrap has both ends of the wrapping wire connected to the oscilloscope.

3.2 Acquisition Requirements

The required acquisition characteristics depend on both the target under attack, and the type of attack being carried out. Considerations with regards to the target under attack include the clock frequency, whether the cryptographic algorithm is in hardware (HW) or software (SW), technology used for the chip, and whether or not countermeasures have been implemented. Typically the capture oscilloscope achieves around 1 GS/s, as shown in Table 3.2.

By measuring the power consumed by a digital device on each clock cycle, it is possible to infer something about the data being processed by this device. This was demonstrated as a method of breaking cryptographic cores using Differential Power Analysis (DPA)[72]. Such measurements are typically done with standard oscilloscopes, which depending on the attack algorithm and device under attack may range from simple low-cost oscilloscopes to high-end specialist oscilloscopes. But if

Table 3.2: A few examples of capture rates in recently published papers. Sample rates only appear if the tested attack was successful at that sample rate.

Reference	Sample Rate(s) - MS/s	Target Type	Probe Type
[44]	5000	HW - 24 MHz	Shunt
[39]	500, 2000	HW	EM
[26]	200	SW	EM
[82]	125, 250, 500	SW - 24 MHz	EM
[127]	500, 1000	HW	EM

the underlying objective is to measure data on the clock edges of the system clock, sampling at the clock rate of the system is sufficient, provided such samples occur at the correct moment (i.e. on the clock edge). This sampling technique is called synchronous sampling, where the sample clock is synchronized to the device clock. A demonstration of this technique to attack the SASEBO-GII board will be given in this chapter (previously published by myself in [101]), where sampling at 96 MS/s synchronously achieves similar results to 2 GS/s asynchronously.

3.2.1 External Clock Inputs

Commercial oscilloscopes typically provide their own sampling clock which is not synchronized to the device clock. In many devices, however, the device clock is readily available either as a digital signal or by adding a buffer circuit to the crystal oscillator. The sample clock can be derived from the device clock to measure a consistent point; for example it can be used to measure the power consumption on the clock edge. A comparison of measurements taken with an unsynchronized and synchronized sample clock is shown in Fig. 3.6. In section 3.1.1 it will be demonstrated that this synchronized sample clock significantly relaxes the requirement of using a high sample rate for certain attacks.

Note that sample clock synchronization is different from the trigger input that all oscilloscopes provide. With a real-time oscilloscope, the internal sample clock of the oscilloscope will be running at all times, and the sample occurs at the next clock edge after the trigger. Thus even though the oscilloscope is triggered at a repeatable time, there will be some random jitter between when the first sample occurs relative to this trigger for unsynchronized (free-running) sample clocks[1].

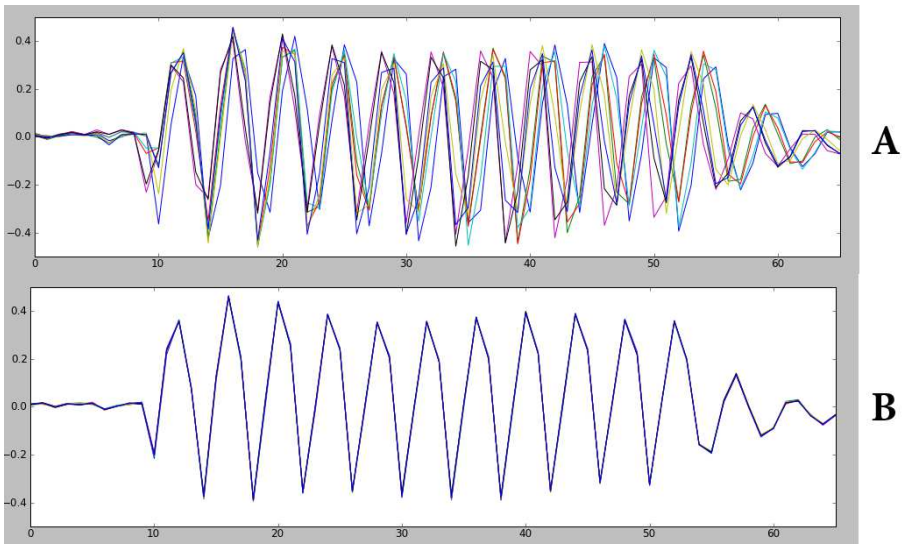

Figure 3.6: Eight power samples with the same input are taken and overlaid to show consistency of measurements. In *A* the sample clock is 100 MHz but not synchronized to the device clock, whereas in *B* the sample clock is 96 MHz, but synchronized with the device clock.

If the clock frequency varies due to either countermeasures or a low-cost oscillator, this would not affect the acquisition quality, since samples are always based on the device clock.

3.2.2 External Clock Phase Adjust

The processing of the external clock input, the ADC, and the analog front-end will add some delay between when the rising clock occurs on the target device, and when the actual sample is recorded. In addition the point of interest for the power analysis may not lie directly on the rising edge, but sometime after this clock edge. For this reason the capture board must be able to add an adjustable delay (phase shift) between the input clock and the actual sample point.

3.2.3 Adjustable Gain

The output of a probe will vary with both the probe type and the circuit under analysis. For this reason, an adjustable gain amplifier is useful to amplify the signal up to the range of the input of the digitizer. Oscilloscopes for example provide a selectable input range - this is still insufficient for H-Field probes, which require an

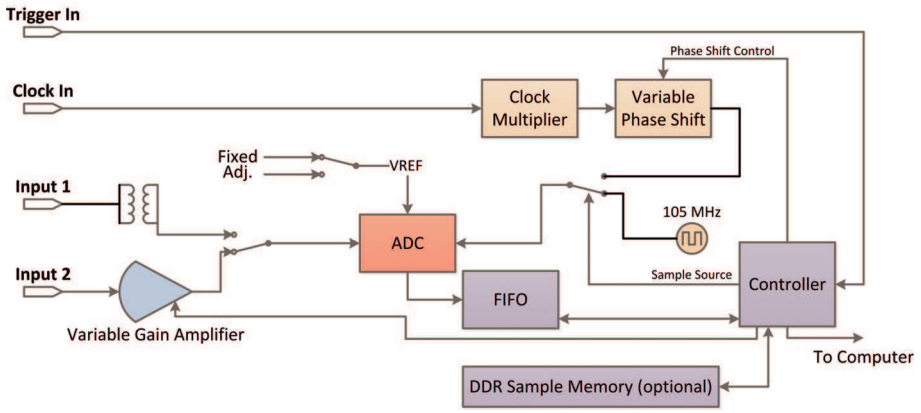

Figure 3.7: Architecture of analog acquisition unit which is implemented in a combination open-source ADC board and COTS FPGA board.

external Low Noise Amplifier (LNA).

3.3 Low-Cost Acquisition Architecture

The architecture of the analog front-end which is used here is shown in Fig. 3.7. The features previously identified as important for side-channel analysis are included: an external clock input with adjustable phase, an internal clock, adjustable gain, and a computer interface.

The analog front-end and ADC board has been released in an open-source design called the OpenADC. The OpenADC hardware consists only of the variable gain low noise amplifier (LNA), ADC, input connectors, and associated support circuitry such as power supplies. This board can be connected to most FPGA development boards with sufficient IO available – it is shown mounted on a low-cost Xilinx Spartan 6 development board from Avnet in Fig. 3.8. The open-source solution includes not only the PCB designs, but example FPGA source code and capture applications on the PC at `http://www.assembla.com/spaces/openadc`. This will form the basis of the ChipWhisperer hardware discussed in Chapter 4.

While the sample rate is limited by the 10-bit ADC selected to 105 MS/s, the analog bandwidth is higher to maintain information on the clock edges. When the LNA input is selected the analog bandwidth is around 110 MHz, and when the transformer-coupled input is selected the analog bandwidth is around 500 MHz. The LNA has an adjustable gain in 100 steps up to 55 dB, allowing for the direct connection of a wide

range of measurement probes, including both current shunt and EM.

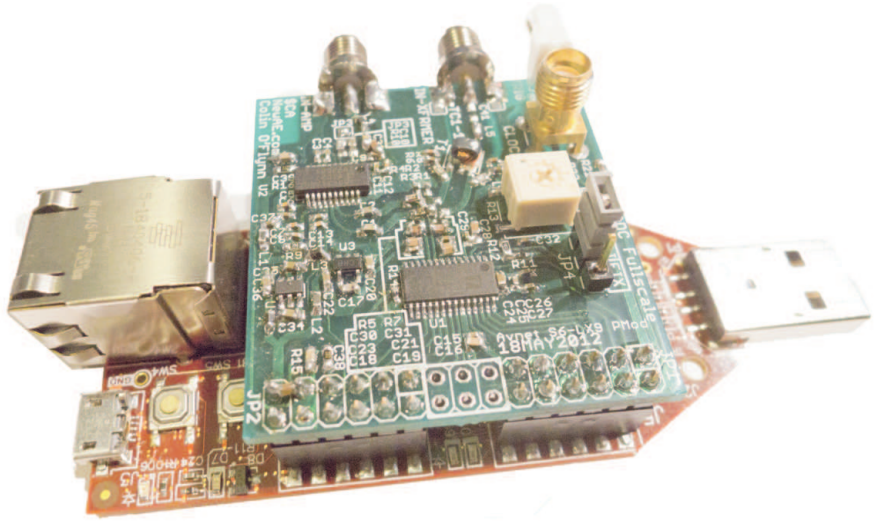

Figure 3.8: The OpenADC mounted on a commercial FPGA development board. The FPGA board provides control, USB interface, and a 48M sample memory.

3.3.1 OpenADC Measurement Results

The results in Fig. 3.9 show that the OpenADC performs well using the inductive wrapping technique. The OpenADC is only sampling at 96 MSPS – but the sampling clock is synchronized to the device clock. When the sampling clock is not synchronized, it fails to recover the encryption key (GSR = 0). This agrees with previously published results on a similar board, which showed a failure of the attack at 100 MS/s [127]. The reference measurement at 2 GS/s is using the oscilloscope's internal timebase; that is to say a timebase that is unsynchronized to the device clock.

The OpenADC has fine granularity on the gain of the input signal, along with the full-scale reference voltage for the ADC. The DSO by comparison does not provide such fine granularity on the input scaling. For the inductive wrap technique it is expected that this partially contributes to the slightly better performance of the OpenADC: the number of bits used to represent the full-scale signal is higher with the OpenADC compared to the DSO, since the OpenADC allows adjustment of the signal to reach closer to the full-scale range of the ADC input.

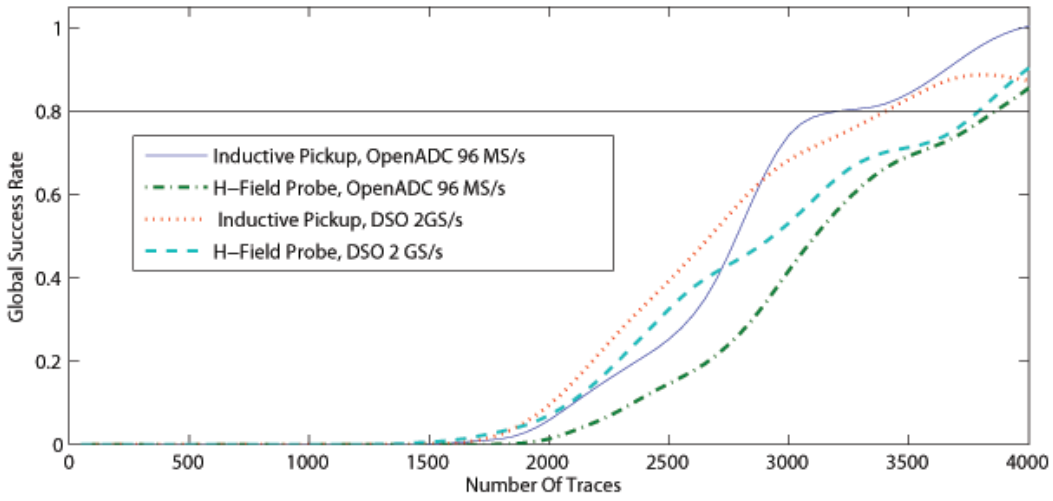

Figure 3.9: Comparison of GSR for traces gathered with the OpenADC and a normal Digital Storage Oscilloscope (DSO), for both H-Field probe and inductive wrapping.

3.4 Summary

This chapter has discussed some of the characteristics of probes and acquisition hardware required for side-channel power analysis measurements.

Rather than using a probe which must be carefully positioned, a probe is built around the decoupling capacitors, which will naturally have most of the high-frequency (e.g.: clock edge) currents flowing through them. It is also trivial to report the measurement setup in a repeatable manner, requiring the following three characteristics: the part number of the decoupling capacitor, type of wire used, and number of wraps around the capacitor.

To acquire the data, it is necessary to use an ADC which is perfectly synchronized to the clock of the device under test. This relaxes the requirement of a high sample rate, allowing low-cost ADCs to be used for side-channel analysis. In addition, the complete design is released as an open-source project, making it available for use by researchers.

There are several main areas of future work to which this capture board can be applied. First, the capture hardware can be extended to support more features. If the device under attack does not provide an accessible clock, a form of 'clock recovery' would be needed, where an adjustable local oscillator is locked to the remote clock (discussed in chapter 5). Secondly, the OpenADC can be used as part of a hardware

implementation of attacks. Attacks could be implemented on the FPGA itself: rather than sending the traces to the computer, they would simply be processed in real-time. This real-time processing would simplify attacks which require a considerable amount of traces, since there is no requirement to store them as an intermediate step.

Chapter 4

ChipWhisperer Attack Platform

This chapter is based on my paper previously published in [105].

The introduction of Differential Power Analysis (DPA)[72] spurned interest in the vulnerabilities of embedded systems previously thought to be secure. The difficulty in comparing results of attacks on different platforms was realized early on, and the SASEBO board aimed to provide a standard platform for attacking [119]. Likewise it was realized that for new entrants into the world of side-channel attacks, having available code and algorithms was a useful starting point such as the OpenSCA toolbox [112], and the DPA Book [80]. Despite this, there is still considerable progress to be made. A researcher looking to replicate existing work, even if that work uses a board such as the SASEBO/SAKURA board, needs to purchase an oscilloscope, interface the oscilloscope to the computer, and (re)implement the attack.

Work into making a complete platform has already been presented, for example the GIAnT system, which uses the same FPGA as this work [108][107] (and with additional details in [67]). Other examples include the FOBOS system [138, 139] which is built with modular FPGA development boards (and can also include the Spartan 6 FPGAs as used in the ChipWhisperer). My ChipWhisperer work was done in parallel to these systems, and the two systems use different architectures and design languages. The ChipWhisperer system presented in this work has more extensive publicly available code for the computer control. Certain features do differ between them: the GIAnT system has a high-speed Digital-to-Analog Converter (DAC) for fault injections of adjustable magnitude, something missing on the current ChipWhisperer hardware. The FOBOS is designed to be used with off-the-shelf FPGA development boards, which avoids requiring custom boards such as the ChipWhisperer depends on.

This work presents a side-channel attack platform which integrates all required

elements: target device, measurement equipment, capture software, and attack software. This work has benefits for almost any user: students have a low-cost laboratory, researchers have an environment which can be duplicated around the world, and embedded engineers have a method of easily testing published research on their own systems. The entire design (both hardware and software) is open-source, encouraging future development from users. Versions of the project are designed to work with existing hardware, such as the SAKURA-G and SASEBO-W board which researchers may already have access to. Modules to control standard oscilloscopes such as PicScopes and VISA-connected devices are also present, encouraging the use of the ChipWhisperer software with existing measurement labs.

Beyond side-channel attacks, the hardware lends itself to glitch and fault attacks. The device runs synchronous to the device under test (DUT), greatly simplifying the introduction of faults on certain clock cycles. A simple glitch generation module is included for inserting glitches at specific offsets from the clock edge.

4.1 Hardware

The hardware consists of both the hardware design and the FPGA code. The system is designed to work with several different FPGA boards, all based on the Spartan 6 FPGA. A 'reference' FPGA board is also provided based on a commercially-available FPGA module, shown in Fig. 4.1. This has a ZTEX FPGA Module with a Spartan 6 LX25 FPGA, however these modules are available in sizes from the LX9 – LX150. Researchers interested in implementing more logic inside the control FPGA may simply switch the ZTEX module for a larger one.

This board provides several features specific to side-channel analysis: two headers for mounting ADC or DAC boards, an AVR programmer, voltage-level translators for the target device, clock inputs, power for a differential probe and Low Noise Amplifier (LNA), external Phase Locked Loop (PLL) for clock recovery, and extension connectors for future improvements such as fault injection hardware. This board will be referred to as the *ChipWhisperer Capture Rev2*.

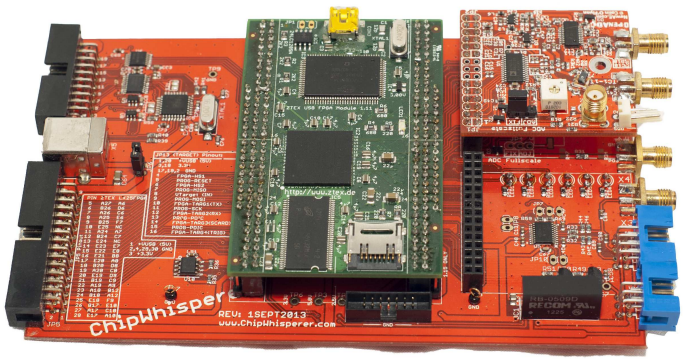

Figure 4.1: The reference implementation runs on a ZTEX Spartan 6 LX25 FPGA Module, with an OpenADC as the analog front-end. The completed board is referred to as the ChipWhisperer Capture Rev2.

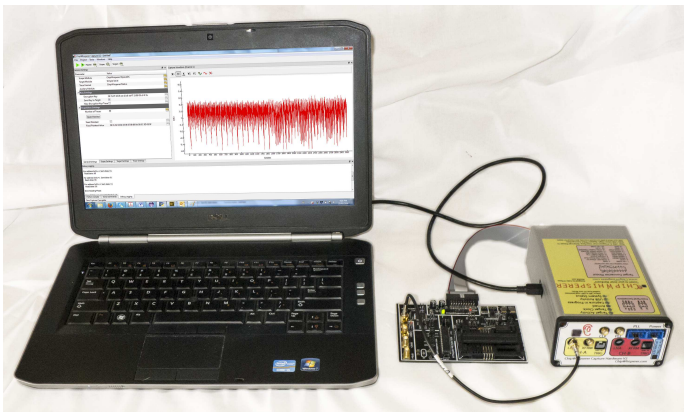

Figure 4.2: The complete system, including the FPGA board from Fig. 4.1 which is mounted in an enclosure, a laptop computer, and the example capture board from Fig. 4.8. The system can also use a breakout board to connect other embedded hardware targets.

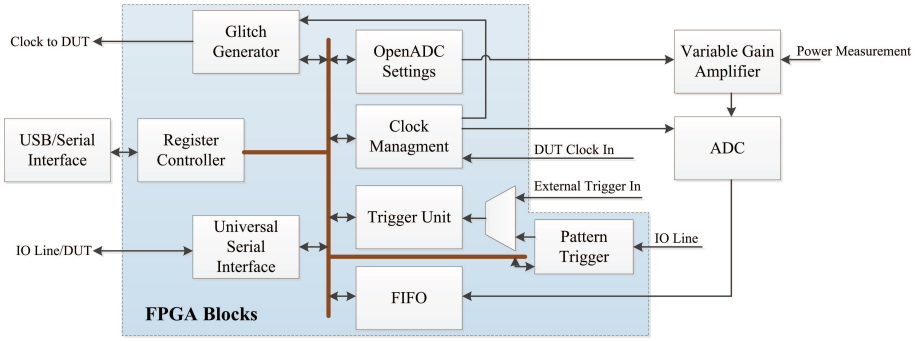

Figure 4.3: The base design consists of several blocks connected via a internal bus in the FPGA, shown in the blue box.

4.1.1 Modular FPGA Design

The blocks within the FPGA are designed around a central 'register control' module, as shown in Fig. 4.3. The design greatly simplifies the addition of new modules: only one small section of the design needs to be modified to insert the bus connections, otherwise the new module can live independently of the rest of the system.

The modular design allows customizing of which modules to include of interest to the researcher; including for example only the clock glitching module if it is desired to work with fault attacks and use a smaller FPGA.

4.1.2 Capture and Clock Control

If the underlying objective is to measure data on the clock edges of the system clock, sampling at the clock rate of the system is sufficient, provided such samples occur at the correct moment (i.e. on the clock edge). This sampling technique is called synchronous sampling, where the sample clock is synchronized to the device clock. Hardware to perform synchronous sampling called the OpenADC was described in [101], where the SASEBO-GII board was attacked, and this demonstrated how sampling at 96 MS/s synchronously achieved similar results to sampling at 2 GS/s asynchronously. The OpenADC is used as the basis for this work.

The analog front-end used here is the OpenADC [101] (also discussed in the previous chapter), which provides a -5 dB to 55 dB gain, simplifying measurement of low-level signals. Additionally designs for a differential probe and Low Noise Amplifier (LNA) are provided, an example shown in Fig. 4.4.

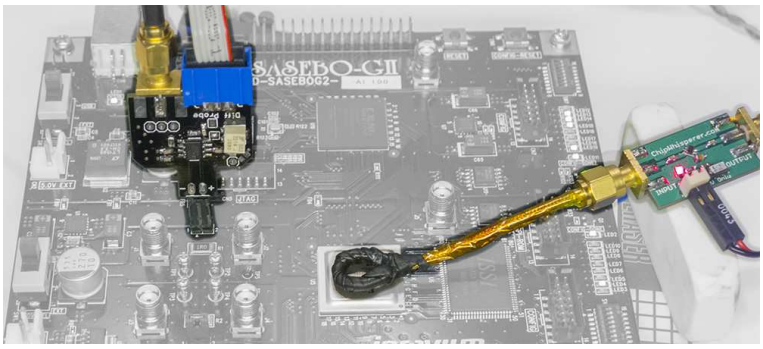

Figure 4.4: Beyond capture hardware, design for a differential probe and H–Field probe with LNA are available.

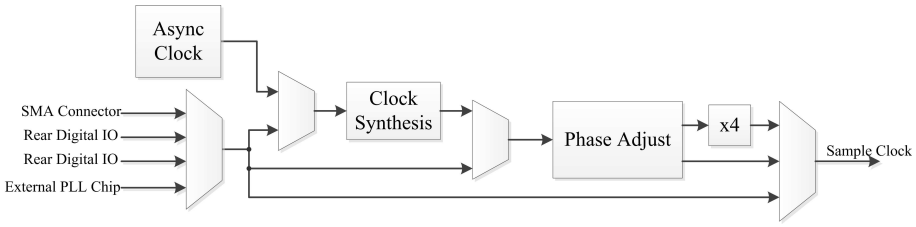

Figure 4.5: Clock Routing in ChipWhisperer capture hardware.

For the synchronous sampling to work, the device must be able to lock onto the system clock. If the clock is readily available as a digital signal (e.g. from the crystal oscillator on the DUT), it can be fed into the FPGA directly, where internally it can be multiplied if desired. If the clock is not available, such as in the case of internal RC oscillators, an external PLL can be used with clock recovery logic to recover the clock (discussed in Chapter 5). Finally an asynchronous clock is available, although due to the limited sample-rate in this platform will have very poor performance compared to synchronous capture [101].

4.1.3 Target Control and Triggering

The FPGA provides some basic IO blocks for driving standard devices. This includes a UART, a Smart Card interface, and Universal Serial IO device, which can be controlled from the computer. Note the target device can be driven by an existing connection instead; the FPGA-based IO blocks are provided simply as a convenience to allow a single USB connection to provide both communications and target control.

Several triggering options are provided. The most basic allows standard triggering: triggering on the rising edge of a digital line for example. This is suitable when analyzing devices which the researcher has programmed, and is able to insert a suitable trigger event into. For more realistic examples, two additional triggering blocks are provided. The first is a digital pattern match, which looks for a specific sequence of transitions on an IO line. This is implemented as a state machine, where it moves through to the next state only if the IO line remains in the expected state for the 'allowed' amount of time. If the IO line fails to match the expected state transition, the state machine resets. This system is specifically designed for triggering on communications protocols, for example by waiting on a response byte. The final triggering system looks for an analog pattern in the waveform, using a Sum of Absolute Difference (SAD) criteria, which is frequently used in video compression, and fast FPGA implementions exist for [106]. Here the system continuously compares the incoming waveform to a known pattern: when the SAD criteria falls below some threshold, the system triggers the capture.

All of these triggering options feature a pretrigger ability. The capture buffer is continuously filled, meaning that the trigger can occur after the actual cryptographic operation has occurred. The limit is simply the size of the capture buffer, which is primarily dependent on the size of the chosen FPGA.

The trigger out signal can be dynamically routed to an external pin. This allows triggering of external equipment with this advanced trigger source.

4.1.4 Glitch Generation

A clock glitch module is also present in the system. Using two adjustable delay lines built into the FPGA, it can insert glitches into a 'target clock': the 'target clock' either coming from the device under test or generated by the FPGA itself.

The glitch width can be adjusted from about 3 nS to 100 nS (maximum width limited to 50% of clock period or 100 nS, whichever is smaller) and the offset from the clock edge is adjustable from -50% to +50% of the clock period. The specific resolution of the glitch and offset varies for the target clock frequency, but is always smaller than 100 pS. Fig. 4.6 shows an example of a glitch inserted into the output clock, although the glitch itself can be output separately from the clock for driving

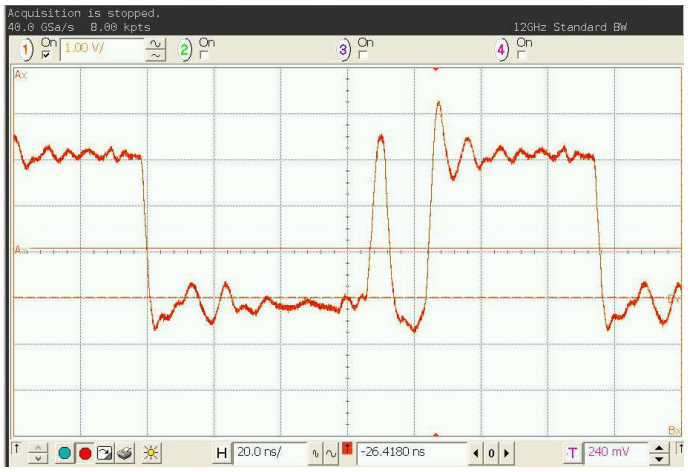

Figure 4.6: Inserting a glitch into a 7.37 MHz clock coming from a target device.

modules such as optical or electromagnetic fault injection. This method is the same as described in [10], although with improved resolution on the glitch width and location. The minimum glitch width is limited by the FPGA IO-pin speed: using a faster IO-standard (e.g. LVDS) allows a smaller minimum glitch width if required.

The use of the special Digital Clock Manager (DCM) blocks along with partial reconfiguration (discussed next) allow this extremely fine-grained control over glitch width. This is an improvement on previously proposed glitch generation methods where the 'coarse' width control comes from a higher-speed clock, which generally limits glitch 'coarse' control to a multiple of this clock speed (often 4–10 nS)[67].

4.1.5 Partial Reconfiguration

The clock generation and glitch generation modules both use the Digital Clock Manager (DCM) blocks within the FPGA. These blocks have limited support for run-time configuration of parameters such as phase delay and frequency generation, and for maximum performance the configuration must be fixed at design time. The Xilinx-provided *run-time* adjustment can shift the phase only by about ±5 nS in 30 pS increments (exact values vary with operating conditions).

To allow adjustments over a wider phase range, a partial reconfiguration interface is provided. This interface allows changes to the FPGA configuration while the system is operating. This is specifically used to reconfigure the DCM blocks for a variety

of parameters which are fixed at the implementation stage. This partial reconfiguration requires that appropriate 'bitstream difference' files are generated by the FPGA tools for every possible setting of the desired parameter, e.g. the DCM phase delay attribute. Due to the opaque nature of the FPGA tools there is no simple mapping between parameter changes and a specific portion of the bitstream.

To generate these bitstream difference files, Xilinx's *FPGA Editor* tool is used to modify the Native Circuit Description (NCD) file for the design. A script generates versions of the NCD file with every possible setting of the desired attribute, e.g. for the DCM block with a fixed phase value of -255 to $+255$. These NCD files are converted into bitstream difference files with the Xilinx *bitgen* utility. Setting the desired DCM fixed phase offset means loading the appropriate bitstream difference file[1].

4.1.6 Implementation on Other Boards

This entire system is implemented as generic FPGA blocks. Whilst a reference platform is provided, it can be used on any FPGA platform. For example this system can be programmed into the control FPGA provided in the SAKURA-G or SASEBO-W platform. Fig. 4.7 shows a photo of the SAKURA-G board with an OpenADC mounted. This system allows implementation of a cryptographic algorithm in the main FPGA on the SAKURA-G, while the control FPGA serves to actually perform the measurements. Any of the available blocks can be inserted into this system, for example adding clock glitch generation into the control FPGA for the SASEBO-W.

4.1.7 Generic Device Under Test Board

For demonstration of basic attacks, a generic target board is provided. This board provides several target options: a 28-pin AVR socket, an XMEGA device, and a Smart Card socket. The board also has two LNAs built onto it, along with several clock options. Jumpers can select which target is connected, measurement mode (high-side or low-side shunt), connect the AVR programmer from the ChipWhisperer Capture Rev2, and allows an external Smart Card reader to be connected to the

[1] See the ChipWhisperer sources for details, with additional information in my article in the June 2014 issue of Circuit Cellar.

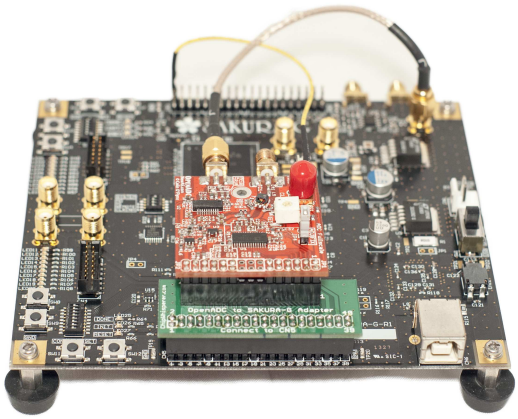

Figure 4.7: The modular design allows easy implementation on other hardware, such as the SAKURA-G board. A Spartan 6 LX75 FPGA is used for a cryptographic algorithm, and the LX9 FPGA is used for control of the FPGA along with capturing of power traces. This replaces both the 'ChipWhisperer Capture Rev2' hardware box and the 'Target Device', meaning the entire side-channel analysis system is present on the SAKURA-G board.

smart card socket by way of a feed-through smart card PCB. The target board is shown in Fig. 4.8.

The choice of a 28-pin AVR socket allows the board to accept many similar AVR devices. Attacks targeting recent processes can use the AtMega328P or AtMega48A. Attacks looking to test older devices may use an older Mega8 device[2]. Note that many 'Smart Card' attacks are tested on a AtMega163 card, which contains an AtMega163 die from Atmel. Existing code targeting the AtMega163 can be ported to work on a 28-pin AVR, avoiding the need to find outdated Mega163 cards, and also using a more recent semiconductor process

4.2 Software Architecture

The software is implemented in entirely in Python. Python was chosen for a variety of reasons: it is natively cross-platform, provides a simple GUI through PySide, can easily interface to other languages including C/C++ and MATLAB, provides high performance using Cython, and has a large collection of modules which provide

[2]While the Mega8 is an older device, recently bought ones may be produced on newer processes. If looking for a device produced on older IC process, one will need to confirm the production date via the date code

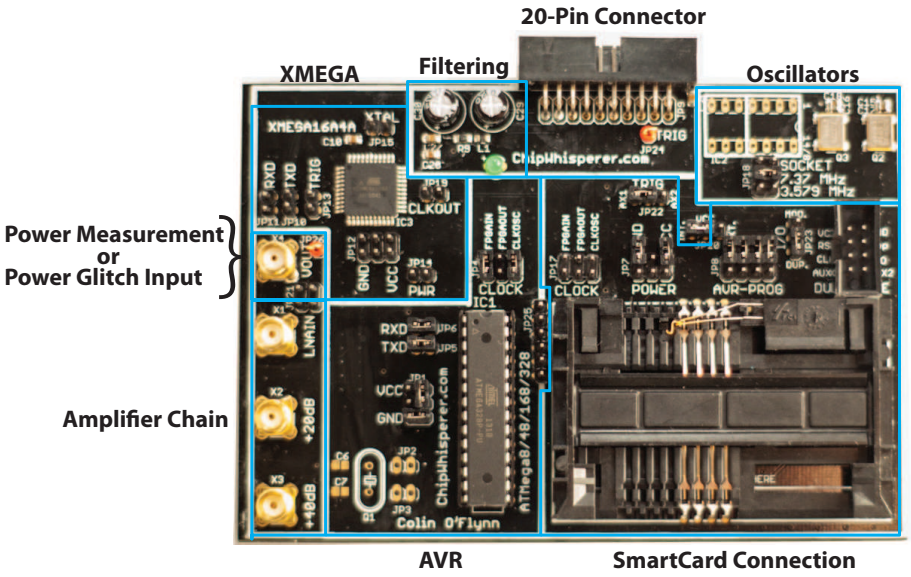

Figure 4.8: The Multi-Target board provides a simple platform for testing various attacks and cryptographic implementations.

functionality such as cryptographic functions, plotting, numeric computations, low-level IO, and smart card interfaces. In addition the choice of an interpreted language such as Python enables considerably more advanced scripting options. This section will only briefly outline the software architecture, full documentation is kept on a wiki at `www.ChipWhisperer.com`.

The project is split into two programs. One program captures the power traces and saves them, the other perform side-channel analysis algorithms (e.g. CPA). The decision to split the project into two programs was done to allow use of only part of this project by other researchers. For example researchers with existing attack code can still use the capture program, for example saving the data to a MATLAB workspace. Or researchers with existing traces can load them into the analysis program without using the capture portion.

Both capture and analysis software share a common base class; this class defines methods of modifying parameters, saving and restoring projects, using trace files, and plotting data. In addition support for a special 'scripting' system is provided. This 'scripting' language allows execution of either the capture or analysis software from another Python application, which forms the 'script'. This design is especially powerful since it allows the script to call any function within the entire program, and

does not require the definition of specific scripting commands. As an example, when adding a new FPGA module, it requires the addition of the appropriate driver module to the capture software. However by using the scripting interface, it is possible to simply send raw commands to this module, which allow testing and debugging it before the complete driver has been written. When options are changed using the GUI, the GUI also shows how to accomplish the same operation with a script. Thus a user can simply setup appropriate options from the GUI, and then save these script commands to recreate the configuration.

The graph windows allow transformations to be performed on the data, such as switching from time-domain to frequency-domain through an FFT, filtering and smoothing, and exporting data in the graph.

4.2.1 Trace Management

The trace management module is common to both the capture and analysis software, which provides a method of mapping traces from different formats into a continuous block of trace data. The default storage method used by the software uses the Python NumPY library's native save and load commands. Alternatives which store the traces to a MySQL server, and saving the traces to the same format used by the DPAContestv3 tools are also provided.

4.3 Capture Software

The capture GUI is ideal for initial experimentations with a new system, such as trying different ADC settings or different trigger settings. For repeatable captures it is desirable to instead script the setup, which sets appropriate values to various parameters. In addition this script can perform actions such as saving the target data to a different format (e.g. a MATLAB workspace), or used for automatically performing captures under different target conditions (e.g. selecting an algorithm with and without countermeasures).

The script can either run the entire capture program, or simply configure part of the window. Several example scripts are provided, which can be loaded in an already-running ChipWhisperer-Capture application.

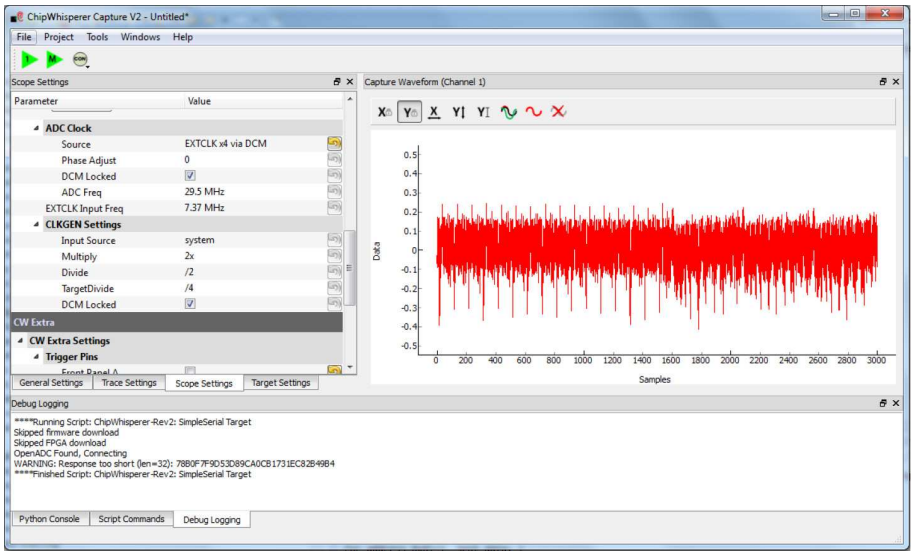

Figure 4.9: The Capture GUI provides an interface to the capture hardware, target device, and storage media.

For debug purposes, a monitor window shows the input and output results of the cryptographic operation being performed on the target. This window also displays what the expected result would be for a known key. This can be quickly used to confirm a device under test is operating as expected, and the encryption key was correctly loaded into the device.

4.3.1 Capture Performance

The capture performance demonstrates how quickly traces can be captured with the default system. The measurements are taken on two different computers: a Windows 7 based Intel i5-2540M laptop, and a Linux based AMD A10-5800K desktop. Captures are averaged over 10,000 traces. The 'Target Connection' indicates how the device under test (e.g. the cryptographic target, *not* the ChipWhisperer capture) connects to the computer. The 'FPGA-x' mode means one of the ChipWhisperer IO blocks are being used.

For high-speed USB targets (ChipWhisperer Rev2, SAKURA-G, SASEBO-W) the capture speed is primarily limited by USB latency in the host computer stack. Note the AVR target shows both 3000 and 20000 points per trace; the resulting speed change is much less than the 6x increase in data size would suggest. Targets connected to the computer directly run much faster, as the IO blocks in the ChipWhisperer tend

to required several USB transactions, each transaction adding latency from the USB stack. This suggests that there is considerable room for speed improvements by streamlining transactions to reduce this latency.

In addition to the OpenADC capture hardware, three standard oscilloscopes are shown for comparison. The capture software can be programmed to support other oscilloscopes with minimal changes.

Table 4.1: Traces/Second for various targets and capture hardware. Points/Trace varies by target, and indicates number of points stored in each trace to attack the target. The '—' indicates lack of supported driver for the host OS.

Capture Hardware	Attack Target	Target Connection	Points /Trace	Traces/Second Win7	Traces/Second Linux
ChipWhisperer Rev2	SASEBO-GII	USB	100	14.8	28.3
ChipWhisperer Rev2	AVR, 38400 Baud	FPGA-Serial	3000	11.3	3.91
ChipWhisperer Rev2	AVR, 38400 Baud	FPGA-Serial	20000	7.04	3.78
ChipWhisperer Rev2	AVR, 38400 Baud	USB-Serial	3000	18.2	18.9
ChipWhisperer Rev2	SmartCard	PS/SC Reader	3000	7.40	6.62
SAKURA-G	SAKURA-G	Integrated	400	6.67	7.18
SASEBO-W	SmartCard	FPGA-USI	3000	0.271	0.279
SASEBO-W	SmartCard	FPGA-SmartCard	3000	1.49	1.52
Agilent MSO54831D	SASEBO-GII	USB	1500	8.01	—
PicoScope 6403D	SASEBO-GII	USB	1500	12.1	43.6
PicoScope 6403D	SAKURA-G	USB	1500	15.4	29.6
PicoScope 5444B	AVR, 38400 Baud	USB-Serial	12000	16.4	5.63

4.4 Analysis Software

The analysis software uses the same project and trace management system used by the capture software. Moving a project over simply means saving the project in the capture software, and opening it in the analysis software. Alternatively, traces can be manually imported, either if they come from an external source or if you wish to combine traces from several different captures into a single analysis run. In addition this manual mode is used for configuring database operation; in this mode the analysis software can read traces from a MySQL database, which allows analysis to occur while the capture is still ongoing.

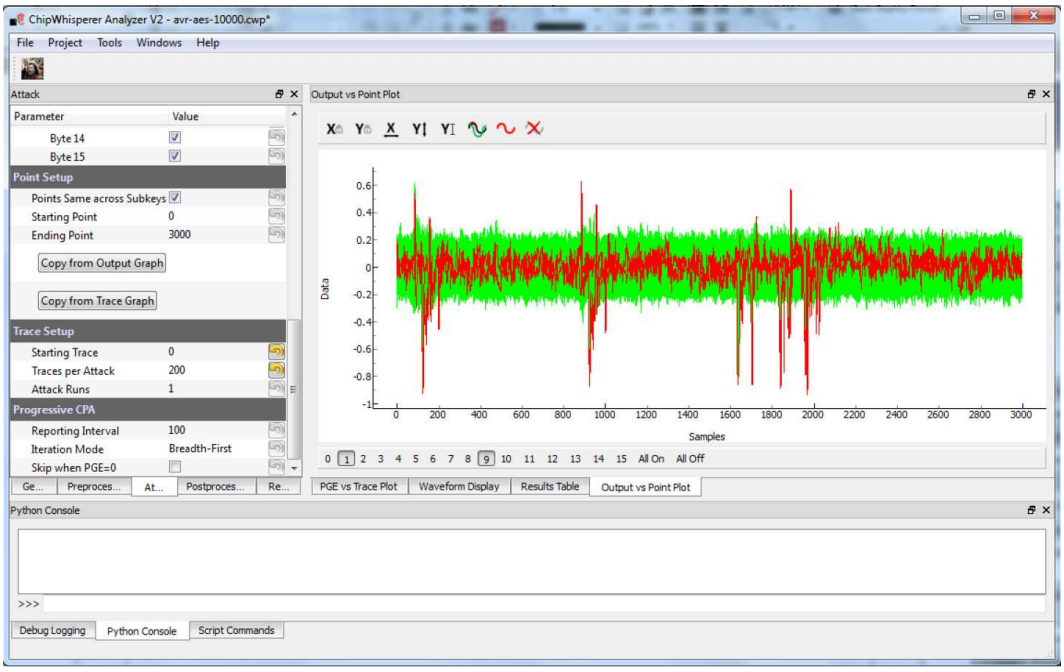

Figure 4.10: The Analyzer GUI runs a given attack on the stored traces.

When traces are loaded, a single trace is plotted in time. A number of traces can be overlaid on each other, which is useful to confirm the synchronization of traces. If traces are incorrectly synchronized, one of the preprocessing modules (described next) can be used to resynchronize the traces.

4.4.1 Preprocessing

Several basic preprocessing modules are provided, which operate on the data before passing through to the attack. Three types of resynchronization are implemented: a sum-of-errors minimizer, peak detect, and cross-correlation. These methods provide very simple sliding resynchronization, which works well with the synchronous capture methodology of the ChipWhisperer Capture hardware. In addition a simple low-pass filter is also provided. Any of the preprocessing modules can be chained together in an arbitrary order, and additional modules can trivially be added to the system.

The waveform display window shows the results after the preprocessing chain. This could be used to confirm that traces are properly resynchronized in the time domain before continuing on to the attack.

4.4.2 Attack Implementation

The attack module is designed to simplify how new attacks are added to the system. The cryptographic model, leakage model, and attack algorithm are all separate modules. This greatly simplifies changes and increases reuse: if a new attack is added, it can pull in the existing cryptographic model and leakage model modules to automatically work with both software and hardware AES implementations. If a new cryptographic model is added such as DES for example, it should work with the existing CPA attack.

The main attack implemented currently is a CPA attack. Fig. 4.11 shows the data flow within the attack system. Data coming from the 'Trace Container' may also have had preprocessing applied, or a certain window of data may be selected instead of the entire trace range. The correlation calculation has several modules that can be loaded, for example selecting between a version that takes advantage of the fast NumPy library, and a version compiled in C using Cython. The 'correlation calculation' may actually implement more advanced algorithms, an example is the Bayesian calculation given in [140] is also implemented. The results are stored in the 'Attack Statistics', which stores results after a given number of traces, and also calculates metrics such as Partial Guessing Entropy (PGE) at the current state [128].

The number of traces to use in the attack is also configurable, and allows for looping the attack several times over different sections of the traces. If 100,000 traces are present in the project for example, one could perform the attack with 1000 traces, and repeat it 100 times. When final metrics are calculated, the system can average the results of all 100 attacks.

Correlation Power Analysis

The basic equation for a Correlation Power Analysis (CPA) attack, where $r_{i,j}$ is the correlation coefficient at point j for hypothesis i, $t_{d,j}$ is power measurement of trace number d at point j, and $h_{d,i}$ is the hypothetical power consumption of hypothesis i for trace number d, with a total of D traces is given in equation 4.1. This basic formula does not allow online calculation, where new traces are easily added without recalculation of the entire sum. Instead the form shown in equation 4.2 is used [22]. This form lends itself to online calculation, where when a new trace is added the sums

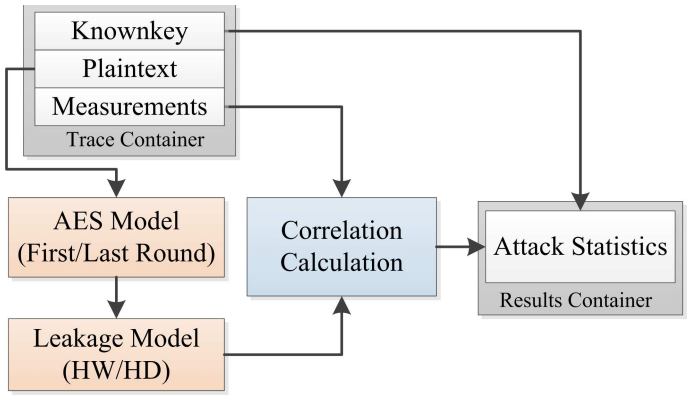

Figure 4.11: In the CPA attack module, the cryptographic model, hardware model, and statistics update are all separate. This allows simple selecting of Hamming-Weight (HW) or Hamming-Distance (HD) models for example, or selecting the AES round to attack.

are updated and the new correlation coefficient calculated.

The online update is used during calculation of attack statistics, where it is desired to track the attack results as new traces are added. As a practical matter, note the denominator of either equation 4.1 or equation 4.2 may have numeric stability problems due to cancellation in either of the two terms. Forms given in [29] may result in more stable calculations, however experimental results shown that for measurements of real systems the numerical stability is not an issue.

$$r_{i,j} = \frac{\sum_{d=1}^{D} \left[\left(h_{d,i} - \overline{h_i} \right) \left(t_{d,j} - \overline{t_j} \right) \right]}{\sqrt{\sum_{d=1}^{D} \left(h_{d,i} - \overline{h_i} \right)^2 \sum_{d=1}^{D} \left(t_{d,j} - \overline{t_j} \right)^2}} \tag{4.1}$$

$$r_{i,j} = \frac{D \sum_{d=1}^{D} h_{d,i} t_{d,j} - \sum_{d=1}^{D} h_{d,i} \sum_{d=1}^{D} t_{d,j}}{\sqrt{\left(\left(\sum_{d=1}^{D} h_{d,i} \right)^2 - D \sum_{d=1}^{D} h_{d,i}^2 \right) \left(\left(\sum_{d=1}^{D} t_{d,j} \right)^2 - D \sum_{d=1}^{D} t_{d,j}^2 \right)}} \tag{4.2}$$

4.4.3 Results Display

The output of the attack is stored in a 'Results Container' type. Generally the assumption is that the output of the attack container will contain a metric for each hypothesis of the subkey, at each point in time. The metric can be sorted to give the most likely subkey hypothesis.

The results display module can plot the value of the attack output for each point

in time, for each of the hypothetical keys. If the correct key is known, this is plotted in a separate color as in Fig. 4.10. In addition if the correct key is known additional metrics can be calculated, such as the Partial Guessing Entropy (PGE). As previous mentioned this PGE will be averaged over many trials when available.

4.5 Example Results

This section demonstrates some example results of the platform. Two example devices will be tested: the first is the AtMega328P microcontroller, which is a reasonably recent AVR microcontroller from Atmel. The system is loaded with basic code which performs encryptions when requested over a serial protocol, and returns the encryption result. The Partial Guessing Entropy (PGE) of the CPA attack is shown in Fig. 4.13 using the *ChipWhisperer Capture Rev2* hardware (i.e. as in Fig. 4.2). As a comparison Fig. 4.14 shows the PGE of the same attack where traces have been recorded with a normal oscilloscope.

This device can also be targeted for glitch attacks. When the AVR was running at 7.3728 MHz, glitches were inserted into the clock with the following specifications: glitch width of 15.2% (20.6 nS), glitch offset of -17.0% (-23.1 nS). The glitch was XOR'd with the clock, and repeated on 200 consecutive clock edges. The objective was simply to cause the embedded system to skip authentication code, which was successfully accomplished.

A second example uses the SASEBO-GII board running at 24 MHz, with the AES core loaded from the 'DPA Contest V3', and the results plotted in Fig. 4.12. Comparison to previously published results from the SASEBO-GII board indicate the ChipWhisperer system is performing as expected[101].

4.6 Summary

This chapter has demonstrated a embedded security analysis platform, which is completely self-contained and requires no additional hardware or software besides a standard computer. The design is extremely modular and allows users to use only a portion of the design; for example using a normal oscilloscope with this system, taking advantage of the advanced triggering mechanisms or the clock glitching capability

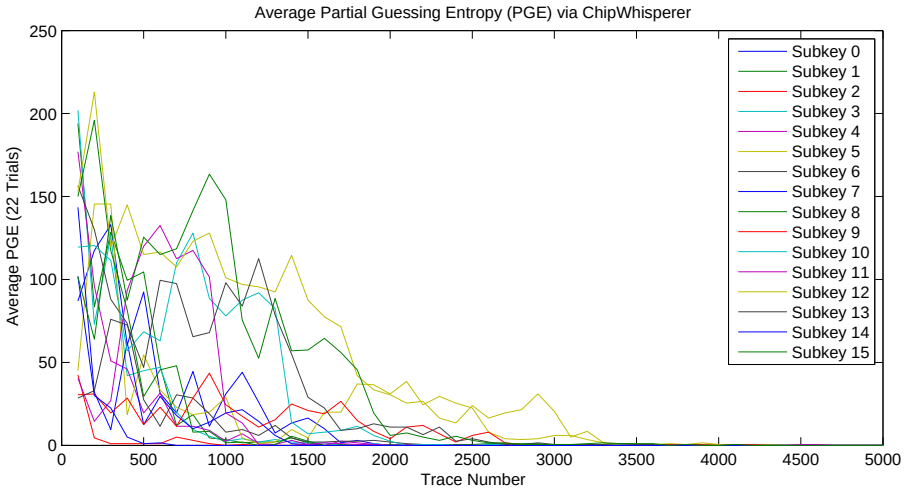

Figure 4.12: Partial Guessing Entropy (PGE) of SASEBO-GII running at 24 MHz. No smoothing has been applied.

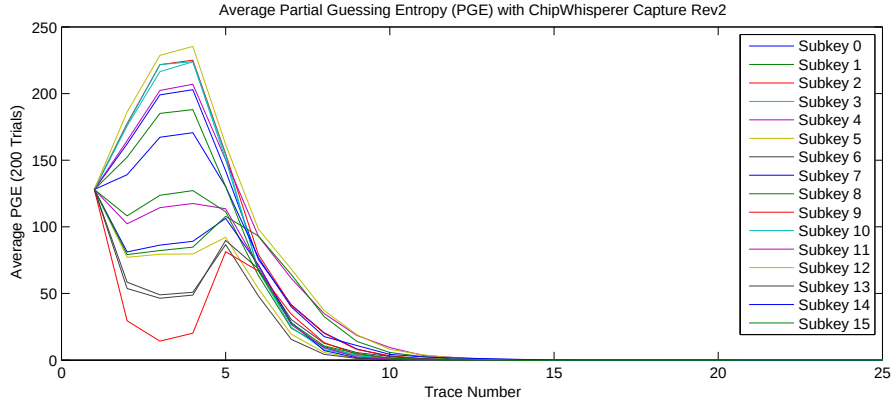

Figure 4.13: Partial Guessing Entropy (PGE) of CPA attack against AES-128 running on AVR Microcontroller. Traces recorded with ChipWhisperer Capture Rev2 hardware at 29.5 MS/s synchronous to device clock. No smoothing has been applied, graph comes from ChipWhisperer Analyzer software.

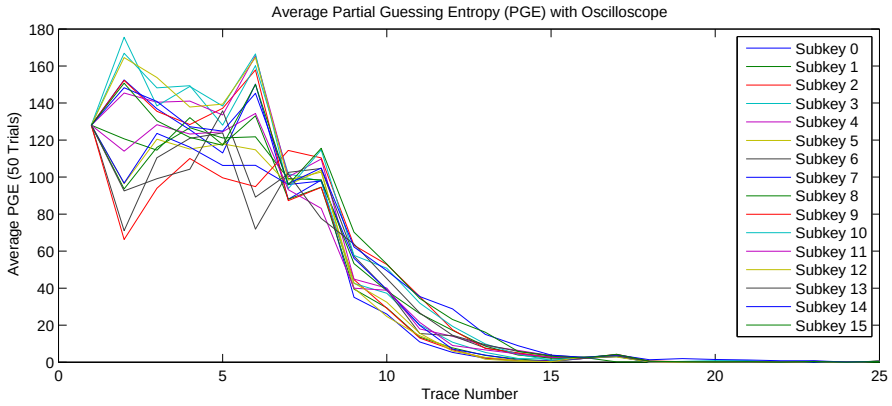

Figure 4.14: Partial Guessing Entropy (PGE) of CPA attack against AES-128 running on AVR Microcontroller. Traces recorded with PicoScope 6403D at 312 MS/s (as with any oscilloscope, this sampling is done asynchronous to device clock). No smoothing has been applied, graph comes from ChipWhisperer Analyzer software.

without using the analog capture hardware.

All design material including source code and hardware design files are maintained in a GIT repository at `www.ChipWhisperer.com`. A wiki is used to maintain documentation, and contributions to either documentation or design are welcome. For users interested in the analysis algorithms, large example captures are available as well: a set of 500,000 traces of AES-128 executed on an AtMega328P microcontroller along with 500,000 traces from the SASEBO-GII.

Since the introduction of this work, a number of researchers have made use of the ChipWhisperer system, demonstrating the usefulness of this open-source platform. Papers using the ChipWhisperer platform include [19, 23, 40, 42, 43, 49, 61, 76, 79, 96, 97, 113, 148].

Chapter 5

Clock Recovery

This chapter is based on my paper previously published in [103].

In Chapter 3, it was demonstrated that a synchronous capture architecture is capable of reducing data requirements by reducing required sample rates to achieve the same success rate. For this to be possible, access to the system clock was assumed to be readily available. For many systems this will be the case—an external oscillator or clock drives the digital logic, and it is trivial to tap into this clock. But some devices rely instead on an internal oscillator; there is no clock signal available for synchronous sampling.

In addition devices may purposely vary the frequency of the internal oscillator in an attempt to stop power traces from synchronizing in the time domain, requiring the attacker to resynchronize the traces after capture. The varying clock countermeasure is assumed to be difficult to reverse in most instances. For example it is claimed in [147] that varying the clock frequency "makes time correlation, a very important step in power analysis attacks, impossible."

If the data was captured asynchronously (i.e. with a normal oscilloscope) with sufficient sample rate, it's possible to compensate for the varying clock frequency via post-processing. This is of little use for attacks requiring real-time information: a trigger matching an analog pattern in the power data, or the injection of glitches timed to specific events requires real-time knowledge of the device clock.

This chapter addresses the problem of recovering the clock from a device under test for both side-channel analysis and fault injection. First, an introduction to the reference platform being used is given, along with a comparison of the synchronous sampling technique to standard asynchronous sampling on this platform.

The platform is then changed to use an internal oscillator which actively varies the frequency during cryptographic operations. Attacks using standard asynchronous

oscilloscopes without preprocessing, with preprocessing, and synchronous sampling are all compared.

Finally a method of performing clock recovery, and using that clock for synchronous sampling is demonstrated. The clock recovery method can be seen as a hardware implementation of the software preprocessing technique. The use of Sum-of-Absolute Difference (SAD) triggers to detect specific events in the system is demonstrated, and finally the injection of glitches is performed on the target while the operating frequency varies.

5.1 Experimental Platform

The device under test (DUT) is an Atmel AtMega48A microcontroller in 28-pin DIP. This device was selected due to several clocking features: it can use an internal or external clock source, the internal oscillator can be adjusted by firmware running on the microcontroller during operation, and the internal clock can be output onto an I/O pin. The differential voltage is measured across a shunt inserted into the VCC pin of the microcontroller. For asynchronous sampling a PicoScope 6403D oscilloscope is used, and for synchronous sampling the ChipWhisperer is used. Full details of the capture hardware and software are available in [105] and at the ChipWhisperer wiki[1]. See Fig. 5.18 for a photo of the test setup.

The 'A' suffix for the AtMega48A indicates it is using a recent fabrication process; the older AtMega48P by comparison is made with a larger (0.35μm) process. The AtMega48P draws more power, and thus would be expected to give a stronger signal across the resistive shunt used to measure current. The AtMega48A thus reflects a reasonable platform which can be compared against any recent digital IC[2].

The crypto module under attack is a C implementation of the AES-128 algorithm. The specific C implementation chosen was 'AES in C' available from avrcryptolib[3]. The attack algorithm is a standard Correlation Power Analysis (CPA) attack[22].

[1] www.chipwhisperer.com

[2] The feature size of this specific device is unknown, but based on similar devices is assumed to be within the 0.12 μm – 0.18 μm range

[3] http://avrcryptolib.das-labor.org

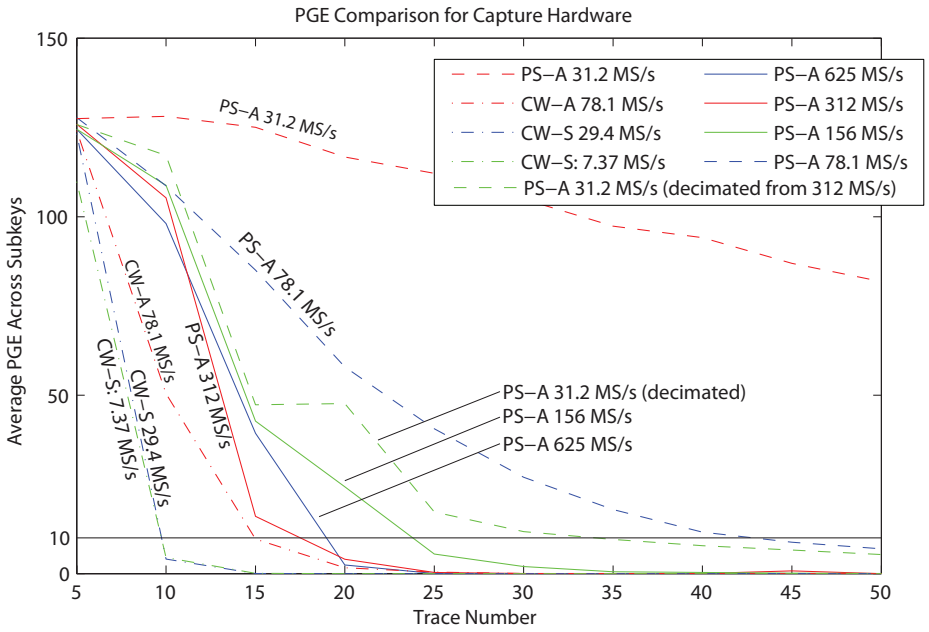

Figure 5.1: PS–A means the PicoScope 6403D sampling at the given sample rate, asynchronous to the device clock. CW–A means the ChipWhisperer in asynchronous mode at the given sample rate, and CW–S means the ChipWhisperer in synchronous mode.

5.1.1 Comparison of Sampling Platforms

While the ChipWhisperer is capable of using both asynchronous and synchronous sampling, it is limited to a maximum sample rate of 100 MS/s. For comparison of higher speed asynchronous captures, a PicoScope 6403D is used for asynchronous sampling, which can achieve up to 5 GS/s.

Fig. 5.1 shows a comparison between the different oscilloscopes and sampling types. For this figure an external 7.37 MHz crystal oscillator was used as a clock source. Results in this chapter will be an average of the partial guessing entropy (PGE) of all subkeys, and where space permits the PGE of each individual subkey is graphed. The reader is referred to Section 2.2 if they are unfamiliar with the PGE metric.

With the PicoScope 6403D (the PS–A data from Fig. 5.1), it is noted that increasing sample rates have improved attack performance initially, but beyond a certain point almost no improvement occurs. For this attack setup there is minimal change from 156 MS/s to 625 MS/s, and in particular the results of 312 MS/s and 625 MS/s are almost indistinguishable.

In the introduction of synchronous sampling, it has been previously claimed the main issue is the random jitter between the trigger event and the first sample occurring that causes the poor performance at lower sample rates in asynchronous systems [85]. We would thus expect a system using a fast sample rate for capture (i.e. so the jitter between the trigger and first sample is minimized), but decimated to a lower sample rate, to have better performance than simply selecting a lower sample rate.

In Fig. 5.1 the line labelled *PS–A 31.2 MS/s (decimated from 312 MS/s)* is captured in such a fashion. The 312 MS/s data is decimated to 31.2 MS/s by selecting every 10^{th} data point and writing them to a new trace file, which the attack is run against. Note that the performance is considerably better than the capture which originally occurred at 31.2 MS/s. No anti-aliasing or other filter has been used in the decimation process. Certain oscilloscopes contain a feature to capture at a high sample rate, and perform such downsampling — the PicoScope 6403D for example provides this option, although this feature is not used in this work.

The performance of the ChipWhisperer hardware at 78.1 MS/s in asynchronous mode shows considerably better performance than the PicoScope 6403D at 78.1 MS/s. It is assumed the built-in Low Noise Amplifier (LNA) in the front-end is resulting in less noise, compared to the PicoScope 6403D which has a more general-purpose front-end.

Finally, note the ChipWhisperer hardware in synchronous mode results in further improvement in performance, despite the considerably reduced sample rates. In synchronous mode the device must sample at a multiple of the 7.37 MHz clock, so sampling is done at 7.37 MS/s and 29.4 MS/s. Both of these results are almost indistinguishable on the graph, indicating that on this particular hardware using a single sample per clock is sufficient.

5.2 Varying Clock Frequency

When an attacker is recording the power traces, ideally each trace would be perfectly synchronized with each other. That is to say that each time instance across all traces corresponds to the same instruction occurring on the DUT. In real systems, traces may not be perfectly synchronized. This could come from jitter in the trigger signal, unintended non-linear code flow such as interrupts on the DUT, or countermeasures

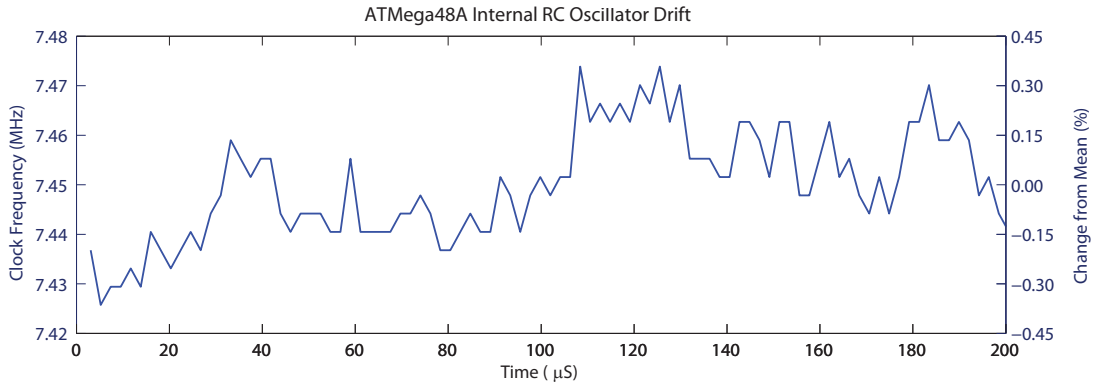

Figure 5.2: Atmel AtMega48A internal clock drift during a side-channel attack.

such as instruction shuffling or random delay insertion. A discussion of algorithms and their performance for resynchronizing is compared in [57]. For all these events the clock is operating at a constant frequency.

Another class of synchronization aims to compensate for the clock frequency of the device varying (called *varying clock* or VC), either due to countermeasures or simply due to the oscillator drift. For an example of the natural variation see Fig. 5.2, which was measured the short-term drift of the internal oscillator on the experimental platform used here. This small amount of variance was enough to prevent the same CPA attack from being successful with over 2500 traces[4], when with a stable crystal oscillator it was successful in only 30 traces. Algorithms which aim to reverse the VC are given in [136, 66, 115, 135].

When a large number of points are required per trace or a large offset from the trigger to the points of interest exist, even the short-term drift differences between the oscillator in the DUT and the oscillator in the oscilloscope may result in desynchronized traces.

With synchronous sampling, variations in clock frequency will naturally be eliminated from the data source. Each sample no longer corresponds to a time instant, but instead to a clock transition. Synchronization may be required for reasons previously discussed such as trigger jitter or countermeasures, but is not needed to compensate for the clock frequency changing.

[4]After 2500 traces the average PGE was 40, and only 4 of the 16 bytes had a stable PGE < 5

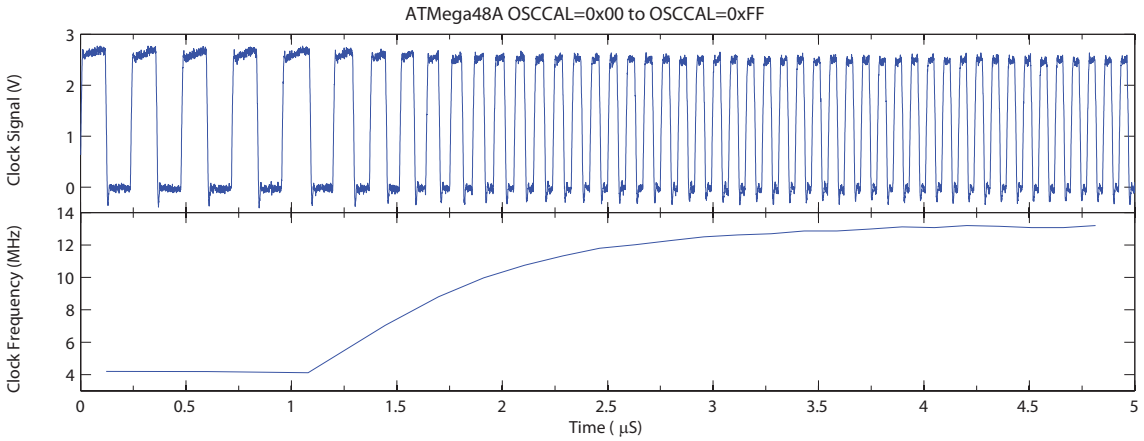

Figure 5.3: Atmel AtMega48A internal clock frequency change as OSCCAL changes from 0 to 255.

5.2.1 Synchronous Sampling of Varying Clock

As a demonstration of synchronous sampling under VC conditions, the AtMega48A target was designed to randomly vary the internal clock frequency before calling the AES encryption routines, and a side-channel attack was mounted. For this initial test the CLKOUT fuse was programmed to output the internal clock onto an IO pin, and the sampling is done synchronous to this clock.

Internal Oscillator Adjustment Range

The AtMega48A datasheet guarantees the oscillator can be calibrated between 7.3 MHz–8.1 MHz, but the actual range is much larger—the specific part used here had a range of 3.95 MHz–13.0 MHz. This test is operating the device outside of guaranteed operating range; commercial products would be advised to only use the adjustment over a smaller range. The time required to switch from the two possible extremes of the randomly selected frequencies, 3.9 MHz to 13 MHz, is shown in Fig. 5.3. The datasheet specifies a maximum change of 2% clock cycle period between cycles for an external clock; it is not clear if this rapidly changing internal oscillator would also be subject to these considerations[8]. For this reason a number of NOP instructs are inserted before beginning further processing after changing the OSCCAL register.

Internal Oscillator Ranges Used

In this course of this chapter, three 'ranges' are used for adjustment of the internal oscillator. The first is the *extended* range, as mentioned spans from 3.9 MHz – 13 MHz. A smaller *narrow* range is also used, which limits the adjustment to a level consistent with the datasheet. Finally the *drift* range is also explored, which reflects the natural random variations due to the nature of the internal oscillator in this device. Detailed information about each of those ranges is presented in Table 5.1. To validate the frequency measurement system, the *crystal* range is also included, where a crystal oscillator is used to maintain a perfect clock reference.

In Fig. 5.4, the histogram of operating frequency during the requested encryptions is shown for the 'drift' range. This appears to follow the normal distribution, as would be expected by a process resulting from random noise. In Fig. 5.5 the histogram is shown for the 'extended' operating range. The value written to the adjustment register (OSCCAL) is uniformly random in the range $[0, 255]$. The AtMega48A splits the OSCCAL register into two over-lapping frequency ranges. Furthermore it does not have linear mapping from the OSCCAL register to operating frequency, resulting in a non-standard distribution[8].

5.2.2 Preprocessing of Traces

The power consumption of a digital device is dependent on the frequency of operation, and this follows a linear relationship. For the ATMega48A at 3.3V, the power consumption when moving from 3.9 MHz to 12 MHz goes from 1.7 mA to 3.1 mA[8]. While the power traces will line up in the time domain with synchronous sampling,

Table 5.1: For the ATMega48A, several different clocking options are used. Two of them purposely vary the frequency of the internal oscillator, one uses the internal oscillator without adjustment, and one uses the standard crystal oscillator.

Name	Range(MHz)	Mean(MHz)	Std-Dev
Extended	3.945 – 12.96	7.210	2.190 MHz
Narrow	7.247 – 8.110	7.663	287.5 kHz
Drift	7.315 – 7.413	7.358	11.78 kHz
Crystal	7.373 – 7.373	7.373	5.469 Hz

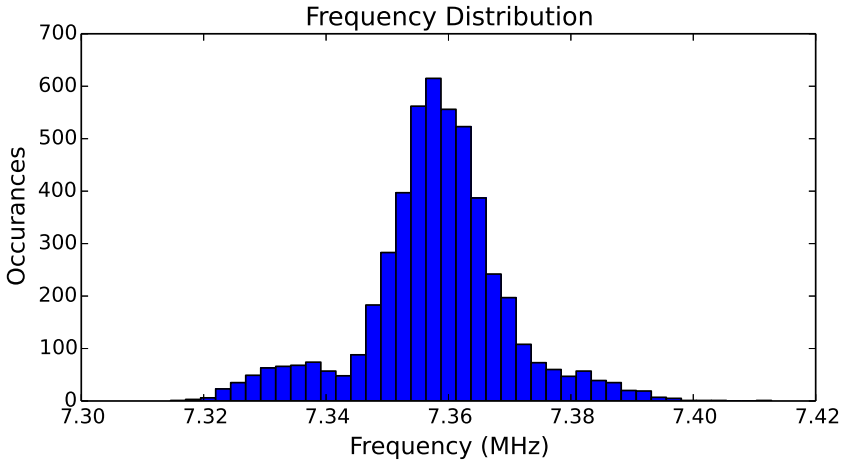

Figure 5.4: The histogram of the operating frequency for the 'drift' range. The distribution appears to approximately follow the Normal distribution.

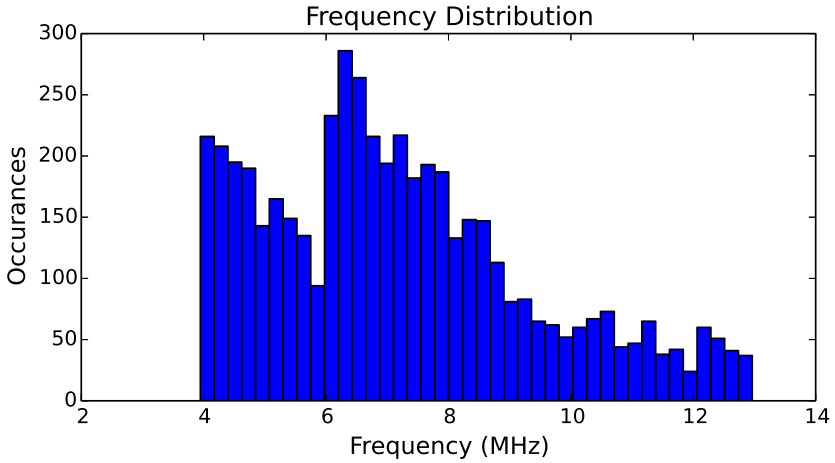

Figure 5.5: The histogram of the operating frequency for the 'extended' range. The non-linear mapping of the control register to operating frequency, which is also split into two overlapping ranges, results in a non-standard distribution.

they will require scaling in order to allow comparison of the same point across multiple traces. In [115] it is suggested to add an adjustment factor based on the measured frequency of operation, as in (5.1). Here $T_{p,n}$ is a single point at index p in trace n, C is a scaling constant, and $f_{p,n}$ is the frequency of the clock at point $T_{p,n}$.

$$T'_{p,n} = T_{p,n} + C f_{p,n} \tag{5.1}$$

This assumes that the change in power measurement due to varying clock frequency simply results in an 'offset' of the measured power. This assumption is also validated in [135], where a 'sliding match' method is used to compensate for the effect of the varying clock on power consumption traces.

To further test this assumption, the mean and standard deviation of each power trace was plotted for the operating frequency f_n, where f_n varies by the 'extended' range given in Table 5.1. The results are shown in Fig. 5.6.

Over a somewhat limited range the assumption appears to hold: for example over the range of approximately 7.2 MHz – 8.1 MHz the mean varies linearly with frequency, and the standard deviation is constant. Thus in this range there is no scaling of values, just a bias which must be corrected for. Over the extended frequency range it would appear some scaling of points is required, as the standard deviation is also varying with frequency.

Four additional preprocessing methods are proposed here; all five methods will be tested by comparing the results of the Correlation Power Analysis (CPA) attack over several frequency ranges.

First, two methods which do not require knowledge of the frequency of operation are proposed. The most basic simply scales all traces to be zero-mean, which again would be expected to only work over a limited frequency range:

$$T'_{p,n} = T_{p,n} - \hat{\mu}_{T_n} \tag{5.2}$$

This can be improved by also scaling by standard-deviation, which should improve performance over a wider range. This will convert the distribution of each trace to be the 'standard normal' distribution. Applying this zero-mean, unit variance normalization (MVN) to side-channel attacks has already been used to improve the

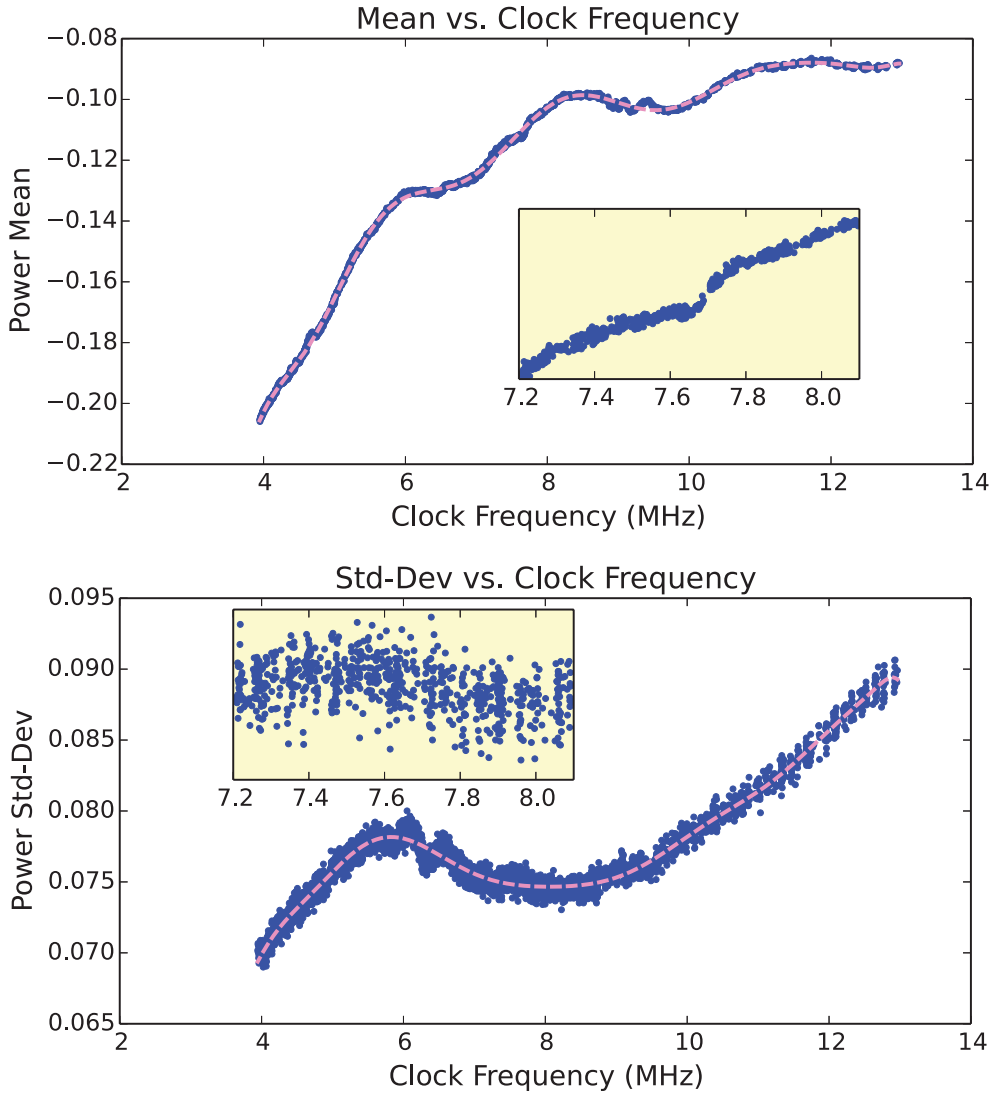

Figure 5.6: Plot of the trace mean and standard deviation compared to operating frequency of the microcontroller, from 3.9 – 13 MHz. Inset details 7.2 – 8.1 MHz range. Dashed line shows the $\hat{\mu}(f)$ and $\hat{\sigma}(f)$ functions used in (5.4) and (5.5).

applicability of template-based attacks beyond the specific hardware which generated the template[88]. This preprocessing is given by:

$$T'_{p,n} = \frac{T_{p,n} - \hat{\mu}_{T_n}}{\hat{\sigma}_{T_n}} \tag{5.3}$$

The main downsides of these methods is they require the frequency be constant over the entire length of the trace. Method (5.1) was proposed in [115] as it could function where the frequency varies per clock cycle. To accomplish this same goal, we will define an estimate function $\hat{\mu}(f)$ which provides an estimate of the mean of the power trace for a known frequency f, and similarly $\hat{\sigma}(f)$ which provides an estimate of standard deviation of the power trace. These functions are simply 15^{th} order polynomial curves fitted to the data in Fig. 5.6. The plots of both functions are shown in Fig. 5.6 as well.

Repeating (5.2) but with $\hat{\mu}$ being a function of f, and not simply calculated over the entire trace:

$$T'_{p,n} = T_{p,n} - \hat{\mu}(f_{p,n}) \tag{5.4}$$

And similarly for (5.3):

$$T'_{p,n} = \frac{T_{p,n} - \hat{\mu}(f_{p,n})}{\hat{\sigma}(f_{p,n})} \tag{5.5}$$

Fig. 5.7 shows traces before and after preprocessing, using (5.5)—note the alignment in the time domain of all the traces due to synchronous sampling, despite the varying clock of the DUT.

Even with synchronous sampling, some trace resynchronization may be required. In this case if the sampling was started and then the clock speed changed, the traces had slight misalignment. It is assumed this comes from either the microcontroller delaying execution during the frequency change, or errors in the sampling ADC as the clock frequency changes. The synchronous sampling still greatly simplified the further resynchronization required, as all traces that required resynchronization required only a shift of 3 or less samples (clock cycles) relative to other traces. If the sampling was started after the clock frequency speed changed, no resynchronization was required, despite the DUT running at different frequencies.

Table 5.2: The number of traces for the Partial Guessing Entropy (PGE) of the CPA attack to be < 10 is given in this table, where the traces have been preprocessed by different methods. $\mathbf{T}_n$ is the unprocessed trace.

Clock	$\mathbf{T}_n$	$\mathbf{T}_n + Cf_n$	$\mathbf{T}_n - \mu_{T_n}$	$\frac{\mathbf{T}_n - \mu_{T_n}}{\sigma_{T_n}}$	$\mathbf{T}_n - \hat{\mu}(f_n)$	$\frac{\mathbf{T}_n - \hat{\mu}(f_n)}{\hat{\sigma}(f_n)}$
Extended	93	32	28	30	28	30
Narrow	23	19	16	15	15	15
Drift	12	12	12	13	12	12
Crystal	29	29	29	30	30	29

5.2.3 Results

The PGE of the CPA attack on an 'extended' frequency variation is shown in Fig. 5.8. Note from Table 5.2 the more widely varying 'extended' range of frequency variation has slightly worse performance than the 'drift' range, thus the varying clock does diminish performance. With the crystal oscillator, performance is similar to the 'extended' range. One would expect it to be similar to the 'drift' range instead, since the frequency is not varying. Thus is assumed to be caused by the external oscillator circuitry in the AVR microcontroller resulting in more noise on the trace measurement. Thus using an internal RC oscillator can actually result in lower-noise measurements compared to an external oscillator.

Attempting to attack anything besides the 'crystal' range with measurements taken by a standard asynchronous oscilloscope fails. The PGE does not significantly improve over the range of trace measurements, even for the 'drift' range. The results of Fig. 5.21 demonstrate this in practice.

As previously mentioned a number of preprocessing methods are also tested, with

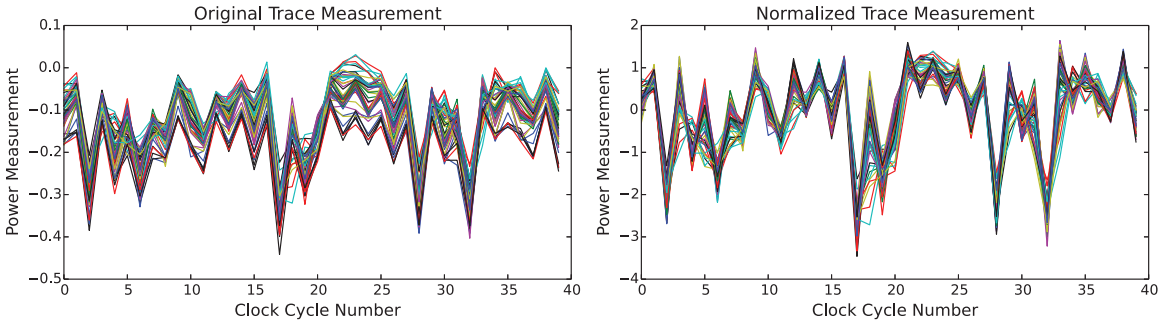

Figure 5.7: Traces can be normalized by (5.5) before passing to a standard CPA attack to remove the effect of varying operating frequency.

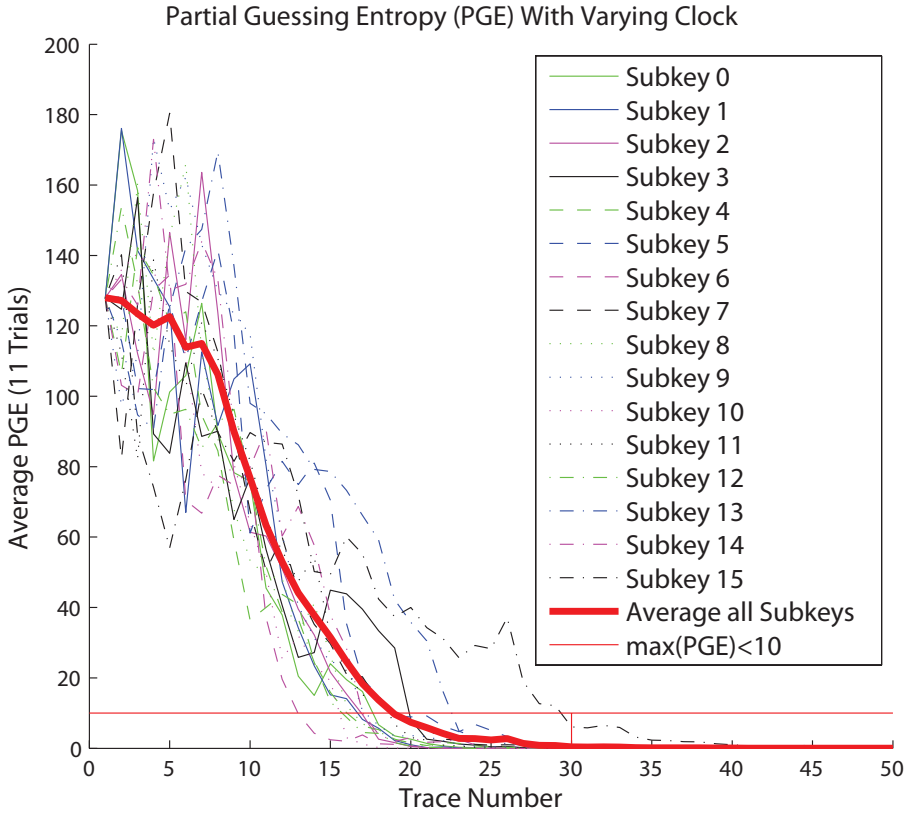

Figure 5.8: Results of a CPA attack on a device with oscillator frequency randomly varying between 3.9 MHz–13 MHz on each encryption, with trace normalization by mean/std-dev, but no trace synchronization being performed. The *Subkey N* refer to the subkey Partial Guessing Entropy(PGE), *Average* refers to the average of all 16 subkeys. $max(PGE) < 10$ shows the metric used in Table 5.2.

final results shown in Table 5.2. The details of the PGE metric are provided in Section 2.1.2. The $max(PGE) < 10$ point is shown in figures as the horizontal line at $PGE = 10$. It can be noted that over a narrow frequency range no preprocessing is required: the 'drift' range has no improvement using any preprocessing method. Only the 'extended' range shows significant improvement in attack performance by using preprocessing, and even then the type of method makes little difference.

These results suggest that details of the preprocessing are not too critical, and would also validate previous work such as [115] which indicate a simple frequency-dependant bias is sufficient. In cases where the frequency is constant over the entire trace, it is sufficient to simply subtract the mean of each trace from itself, forcing the trace to be zero-mean.

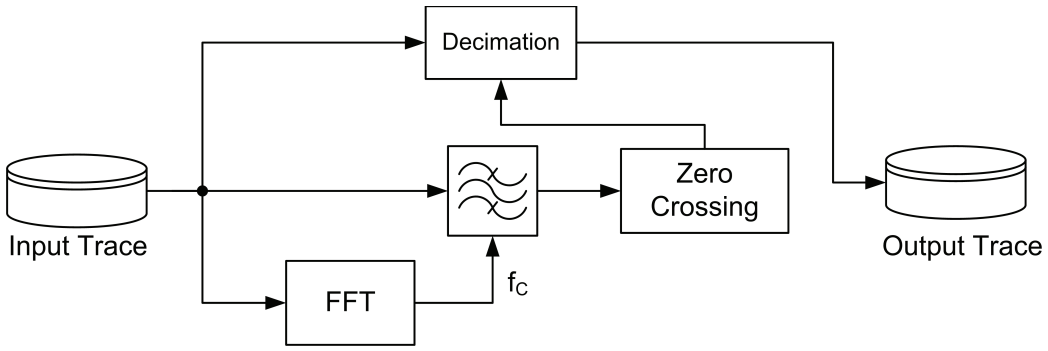

Figure 5.9: Previous work on trace compression can be considered a simple example of clock recovery. Here the trace compression is performed by simply detecting zero-crossing events which correspond to clock edges, and only storing those points.

Considering the extremely large range the oscillator was varied over (3.9 MHz–13 MHz), these results show that synchronous sampling is a simple method of attacking the varying clock (VC) countermeasure.

5.3 Clock Recovery as Preprocessing

If we consider the case of asynchronous sampling, where the sample rate is infinitely fast, the synchronous sampling method would be equivalent to performing trace compression which is keeping a single point per clock sample [80]. Practically of course this means simply sampling 'fast enough' for a specific target; looking at Fig. 5.1, we can see for the AtMega48A sampling at 312 MS/s should be sufficient.

In this work, recovering the clock is done by filtering the recovered signal around the fundamental frequency component. This method is used since it is possible to implement in both software and hardware. In particular the hardware implementation will be used for real-time recovery of a device clock for synchronous sampling and glitch generation.

Fig. 5.9 shows a block diagram of the clock recovery and decimation logic. A FFT is used on the input trace to determine the operating frequency of the device, where it is assumed the operating frequency results in the largest harmonic component. Systems with multiple oscillators may require a more complex selection logic.

A 5^{th} order IIR Butterworth bandpass filter with a center frequency f_C processes

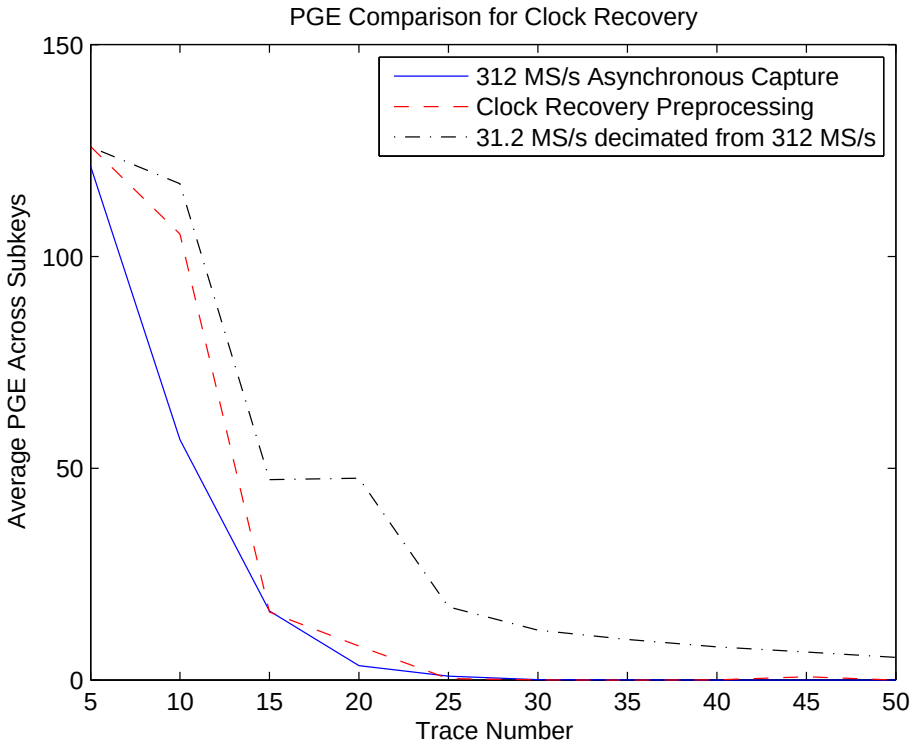

Figure 5.10: All results come from same traces, captured on PicoScope 6403D at 312 MS/s. Each trace in the raw file has 31888 points, the clock recovery version has 1500 points, and the decimated trace file has 3188 points.

the received data, where f_C is selected as the device operating frequency. The passband of the filter is configured to have a bandwidth of 20%, where the bandwidth for a filter with a passband from f_L to f_H is given by (5.6).

$$BW = 2\frac{f_H - f_L}{f_H + f_L} \tag{5.6}$$

The sample corresponding to a clock edge is selected based on a zero-crossing detection of the filtered output. This means the effective sample rate becomes $2\times$ the device clock frequency, since two zero-crossings are produced for every cycle.

The results of a CPA attack against a system where the clock is constant, i.e. the *crystal* range, is shown in Fig. 5.10. This comparison shows that the clock recovery logic can reduced the trace size with minimal impact on attack performance. Using integer decimation by comparison results in a performance penalty on the attack results.

5.3.1 Clock Recovery with Varying Clock

Initially, the internal RC oscillator is used without any explicit random frequency generation. The RC oscillator does randomly drift about ±0.5% during operation as measured in Table 5.1. Measurements taken with a standard oscilloscope fail to recover the key as shown in Fig. 5.21, even after 1000 trace measurements. When the clock is stable the standard oscilloscope recovers the key in < 20 traces, as in Fig. 5.1. Thus the small amount of clock variation causes the CPA attack to fail, despite the starting point having perfect synchronization. If instead we use clock recovery algorithm from Fig. 5.9, the results are as in Fig. 5.11. The CPA attack is successful and with similar success to the original setup.

Next, the 'narrow' frequency range in Table 5.1 is used for clock recovery, which has a center frequency of 7.66 MHz. The frequency was varied approximately ±5.5%. Fig. 5.12 gives the results of the CPA attack on this system.

Finally, the 'extended' clock frequency range which varies from 3.9 MHz – 13 MHz is used, which has a center frequency of 7.21 MHz. Fig. 5.13 gives the results of the CPA on this setup.

These results show that the CPA attack remains successful on all targets, despite the highest operating frequency being over 3× the lowest operating frequency.

5.4 Clock Recovery Hardware

In many devices the clock is not available externally, meaning additional work is required to perform synchronous sampling. In side-channel analysis, it was previously demonstrated how to force an internal oscillator to lock to an external signal [123]. This was used to stabilize the internal RC oscillator and improve trace synchronization, but the same method could be used to generate the reference clock for synchronous sampling. This will fail if the device itself is varying the clock frequency, so instead *clock recovery* must be used to generate a copy of the clock. The idea of clock recovery is not new—in communications electronics this has been used for many years to synchronize a receiver clock to a transmitter clock over long distances [36].

The basic method used here for clock recovery is to filter the power signal so that only the fundamental frequency from the internal oscillator is left. This can then be

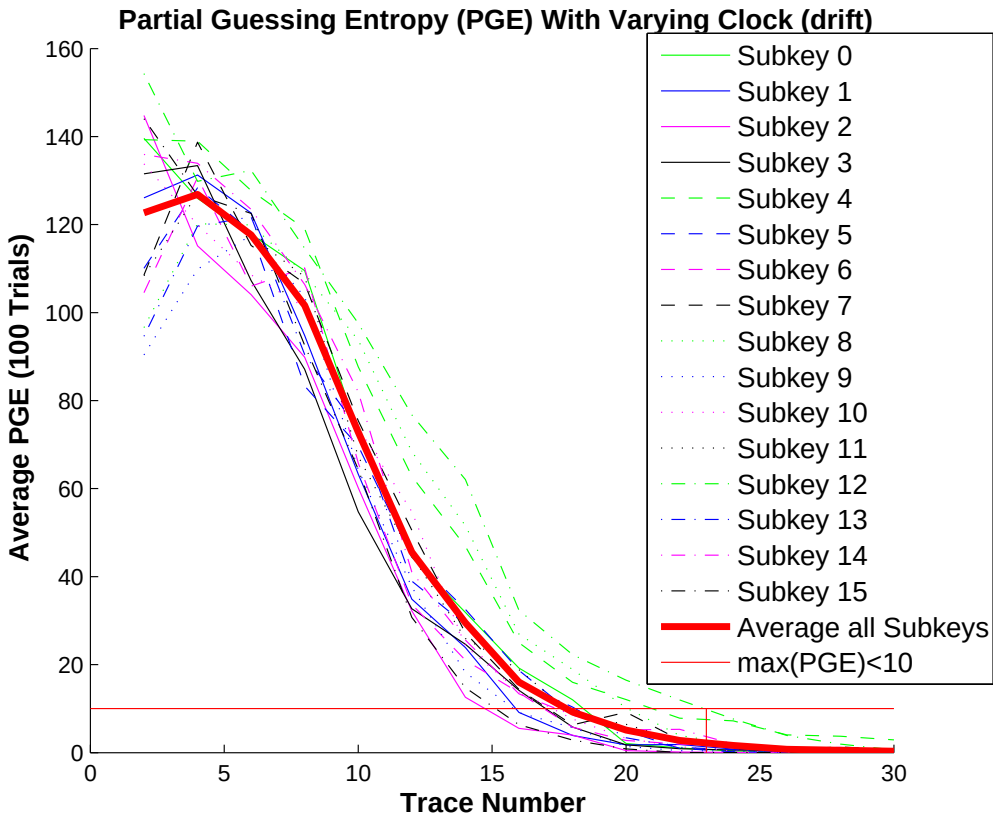

Figure 5.11: Results of a CPA attack on a device with an internal RC oscillator, where the oscillator frequency changes $\pm 0.5\%$ during operation due to drift, and the clock is not externally available, but clock recovery as a preprocessing is used. *Average* refers to the average of all 16 subkeys. Subkey plot legend same as in Fig. 5.8.

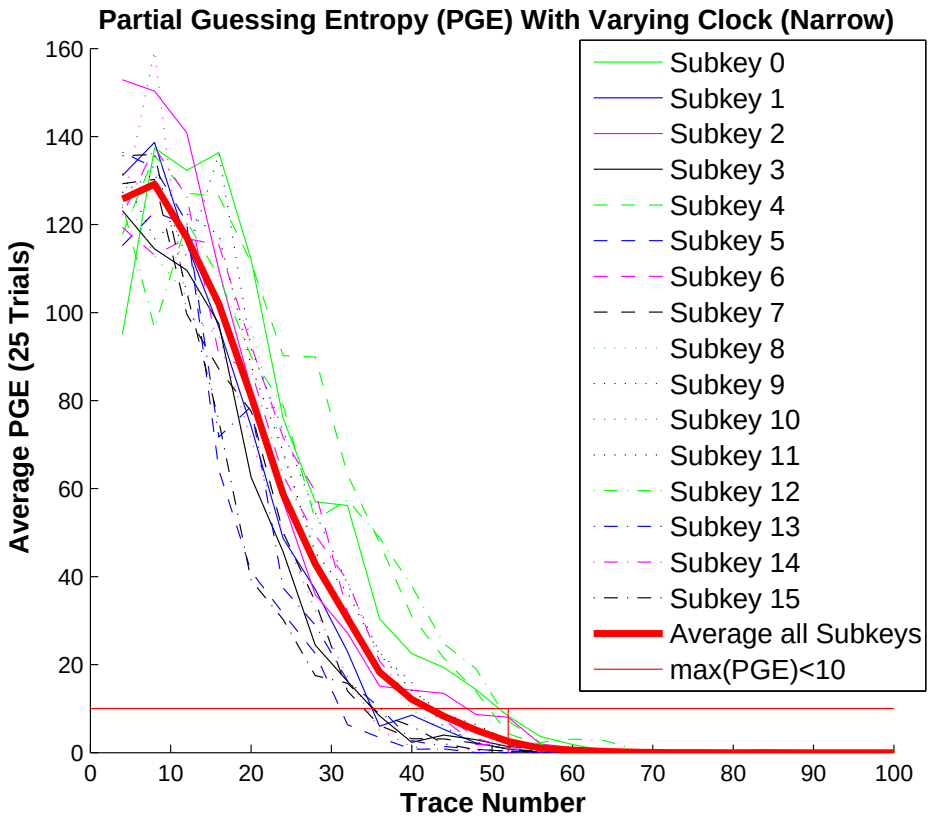

Figure 5.12: Results of a CPA attack on a device with an internal RC oscillator, where the oscillator frequency changes ±5.5% during operation, and the clock is not externally available, but clock recovery as a preprocessing is used. *Average* refers to the average of all 16 subkeys. Subkey plot legend same as in Fig. 5.8.

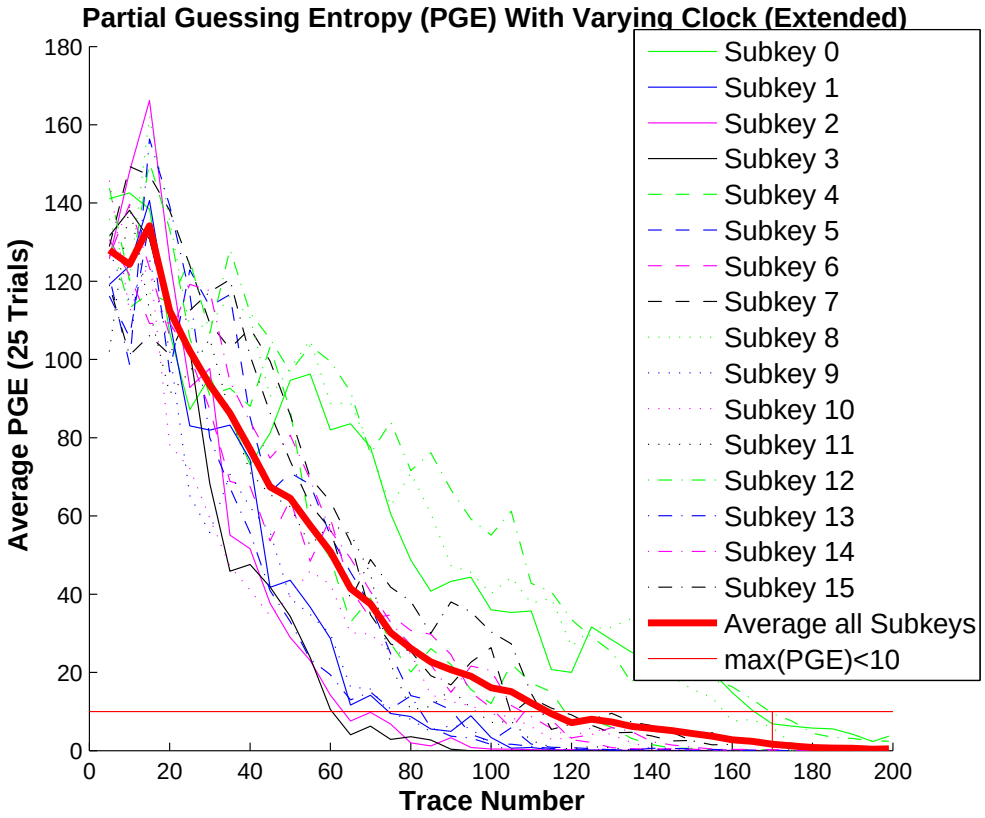

Figure 5.13: Results of a CPA attack on a device with an internal RC oscillator, where the oscillator frequency changes -45% to $+80\%$ during operation, and the clock is not externally available, but clock recovery as a preprocessing is used. *Average* refers to the average of all 16 subkeys. Subkey plot legend same as in Fig. 5.8.

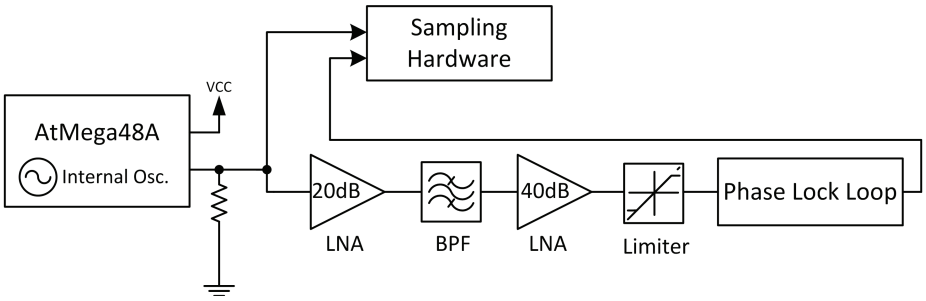

Figure 5.14: Clock recovery block diagram.

amplified and turned into a digital signal. To prevent glitches from resulting at the output a PLL is used to provide a clean digital signal. Details of this hardware design and results of side-channel analysis tests will be presented next.

5.4.1 Hardware Design

A block diagram of the system is given in Fig. 5.14. A Low Noise Amplifier (LNA) is placed on each side of the band-pass filter (BPF), the BPF selecting the fundamental frequency from the power signal. The output of the final LNA is limited to logic levels and fed into the Phase Lock Loop (PLL) block. The PLL used is a single-chip solution, the Texas Instruments CDCE906 device which integrates the Voltage Controller Oscillator (VCO), Phase Detect (PD), loop filters, and frequency dividers into a single package. For an introduction to PLLs the reader is referred to [13].

Fig. 5.15 shows an example of recovering an internal oscillator on an Atmel AVR ATMega48A device. With this device it is possible to switch on a 'clock out' pin, which allows measurement of the internal RC oscillator signal. The clock recovery logic works equally well with this pin enabled or not, but enabling the pin allows comparison of the recovered clock to the internal oscillator.

5.4.2 Filter Design

The design of the band-pass filter (BPF) is critical for the success of the clock recovery. Selection of the pass-band is based on the frequency of the internal oscillator for the device under attack. If this frequency is not known it can typically be found by viewing the frequency spectrum of the device during operation.

Careful consideration must be given for the group delay of the filter, which changes

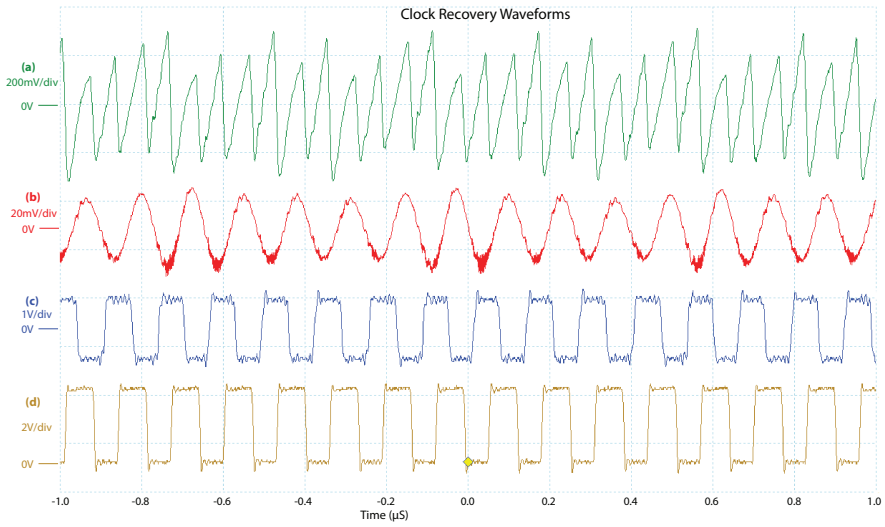

Figure 5.15: Recovery of 7.7 MHz Internal RC Oscillator on ATMega48A. (A) is the amplified power trace after the LNA. (B) is the output of the band-pass filter, and (C) is the output of the limiter, which generates a logic-level signal. The output of (C) can be passed through a PLL to further stabilize the signal. In (D) the actual RC oscillator output is shown, note the perfect alignment of the recovered signal (C) and internal RC oscillator (D).

over frequency. As an example the 6.5 MHz–8.5 MHz BPF used for the ATMega48A device is shown in Fig. 5.16. The group delay, which is usually measured in time units or phase degree, has been scaled by the frequency to give us a group delay in 'clock cycles'. The group delay will cause synchronization errors between traces if the frequency of the DUT oscillator changes, since the delay through the filter varies with frequency.

For more detail, the delay between the actual internal RC oscillator and the recovered clock is plotted in Fig. 5.17 over a more limited range. Here the delay is measured in degrees, where 360 ° equals one clock cycle. This figure comes from measurements of the final implemented system, whereas Fig. 5.16 is based on simulations of just the BPF.

Three methods to reduce this error can be used. First, the type of analog BPF should be matched with the DUT. If the frequency of the oscillator varies only a tiny amount, it would be possible to use a Chebyshev filter with the better attenuation performance. If the DUT oscillator frequency will vary a filter with better group delay performance could be used such as the Bessel. The second way to reduce this

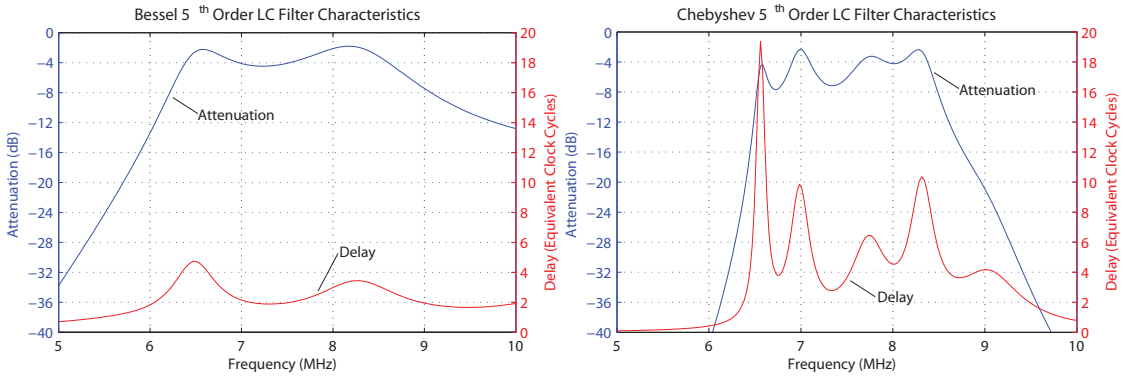

Figure 5.16: Choice of filter type means a choice between better group delay performance and better attenuation outside the pass-band. Two examples are given here: a Chebyshev filter and a Bessel filter, both 5th order made from discrete LC components. Results are from simulation.

error is to measure the frequency during each trace acquisition, and shift the recorded waveform by the known group delay of the filter at this frequency. Finally a standard trace synchronization algorithm can be used to synchronize all such traces.

5.4.3 Results of CPA Attack

The AtMega48A platform is used again for this evaluation. The 'external clock' output is disabled during these tests—the AVR driving the IO pin at the clock frequency results in a very strong fundamental harmonic on the power trace, which results in a better signal for the PLL to lock onto. Such a system would be unrealistic since real systems would not be driving an arbitrary IO pin causing this strong fundamental.

The complete setup with clock recovery module, OpenADC capture hardware, and target is shown in Fig. 5.18.

The test setup is almost identical to that of Section 5.3.1, where clock recovery is done via processing of traces capture asynchronously. Again initially only a small frequency variation due to drift of about $\pm 0.5\%$ during operation is used, as measured in Table 5.1. With synchronous sampling with clock recovery as proposed in this chapter, the results are as in Fig. 5.19. The CPA attack is successful without any special processing of the traces.

Next, the 'narrow' frequency range in Table 5.1 is used for clock recovery, which has a center frequency of 7.66 MHz. Fig. 5.20 gives the results of the CPA attack on

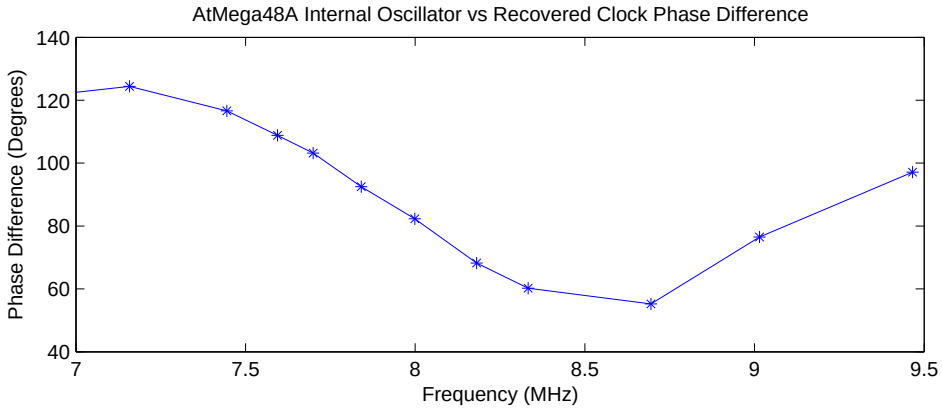

Figure 5.17: As the phase difference changes, the alignment of measurements is compromised, requiring more traces. This figure shows the measured phase difference for the overall system, i.e. phase difference between the RC oscillator on the AVR and the final recovered clock. A Bessel analog filter is used here, results are from measurement.

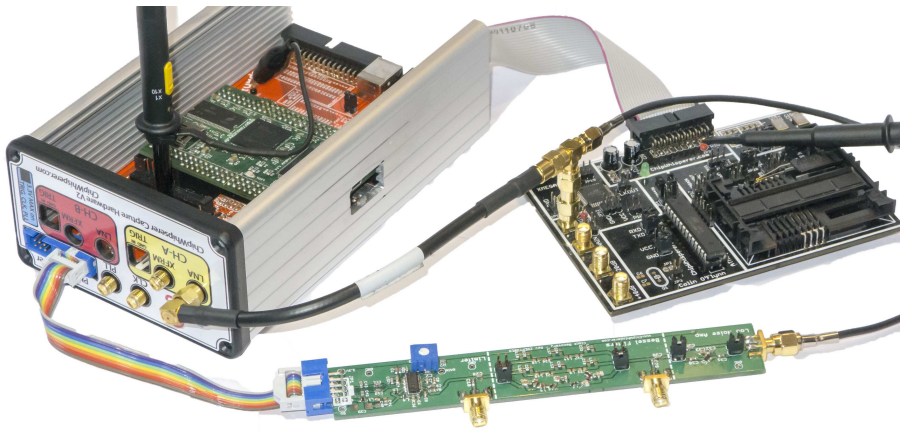

Figure 5.18: Test Setup for side-channel analysis with clock recovery of internal oscillator on ATMega48A. The oscilloscope is used to measure recovered clock frequency. The long board center-front performs amplification, filtering, and limiting. The PLL is located inside the capture hardware on the left-hand side. The back right board is the AtMega48A target.

this system. The reduced performance is mainly due to the phase delay of the clock varying with frequency, as in Fig. 5.17. When the clock is directly available and not obtained through clock recovery, as in the results of Table 5.2, the 'narrow' frequency range has similar performance to the 'drift' range.

The 'extended' clock frequency range of 3.9 MHz – 13 MHz could not be recovered using the simple filtering method. This is due to the fact that the 3^{rd} harmonic of 3.9 MHz will be at 11.7 MHz, which would fall within the bandpass filter bandwidth. Using clock recovery on a very widely varying clock would require a tunable filter which follows the fundamental frequency.

Note that comparing the results to the software-based clock recovery from Section 5.3.1 shows that asynchronous sampling has better performance, it is assumed due to the ability to generate an ideal filter, instead of being limited by physical component selection. The clock recovery method is still useful when it is desired to use synchronous sampling due to the reduced sample rate requirement compared to capturing asynchronously and later processing the data. For fault injection processing the data after capture is not useful, since real-time information is required. The next section will concentrate on the use of clock recovery for these cases.

5.5 Fault Injection

For injecting faults into an embedded system, having a clock which is phase-locked to the device clock allows more precise temporal location selection. If triggering must count a certain number of clock cycles for example, this is difficult to do over long periods due to drift in either the device clock or the instrument clock. If the device clock itself is used, it is trivial to count over a large number of cycles with great accuracy.

Previous work has looked at either disabling the switch to an unstable clock [137], or forcing the internal clock to lock to an external clock[123]. These methods are highly dependant on a specific system design; a device may instead always come up on an internal oscillator, making it impossible to keep it running on the external clock.

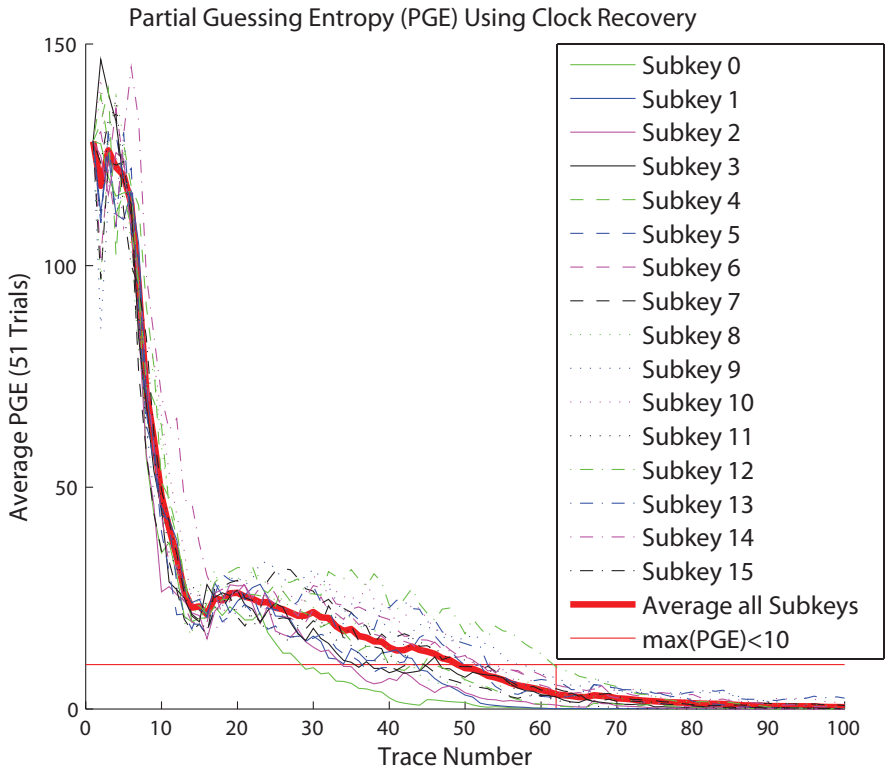

Figure 5.19: Results of a CPA attack on a device with an internal RC oscillator, where the oscillator frequency changes ±0.5% during operation due to drift, and the clock is not externally available, but clock recovery with synchronous sampling used. *Average* refers to the average of all 16 subkeys. Subkey plot legend same as in Fig. 5.8.

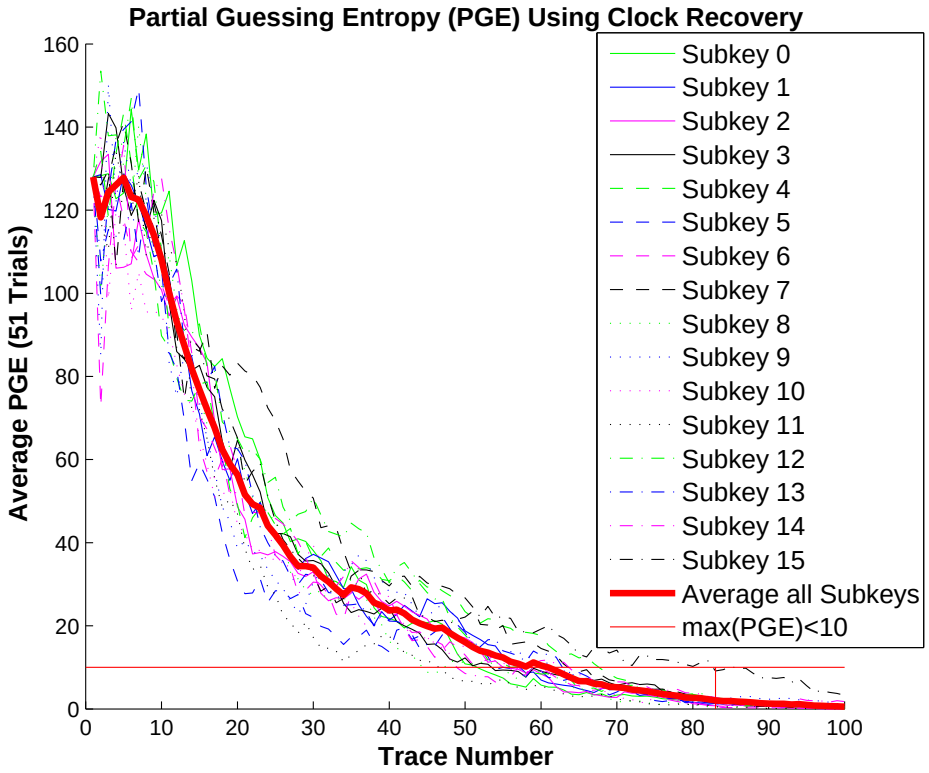

Figure 5.20: Results of a CPA attack on a device with an internal RC oscillator, where the oscillator frequency changes ±5.5% during operation, and the clock is not externally available, but clock recovery with synchronous sampling used. Plot legend same as in Fig. 5.8.

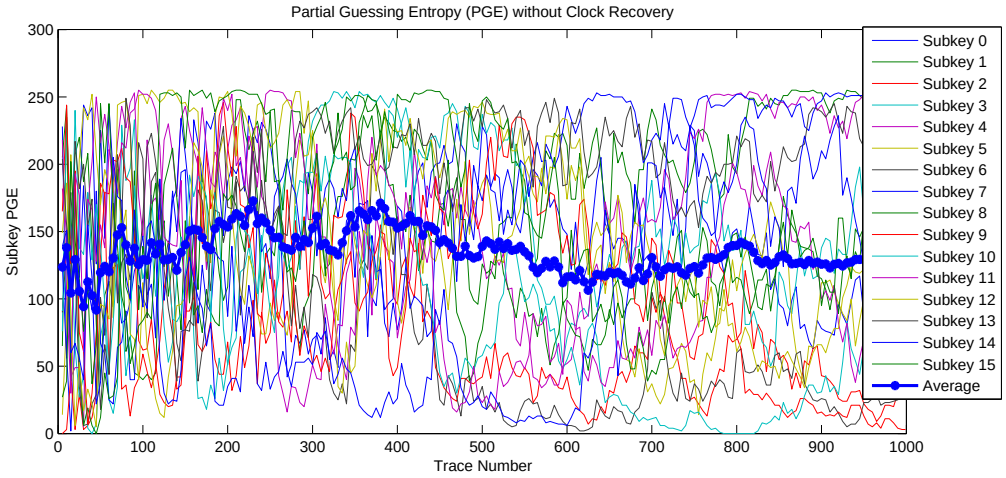

Figure 5.21: Results of a CPA attack on a device with an internal RC oscillator, where the oscillator frequency changes ±0.5% during operation due to drift, and a standard *asyncronous* oscilloscope samples the device at 312 MS/s. *Average* refers to the average of all 16 subkeys.

Having a phase-locked clock means glitches can even be inserted at specific portions of the device clock cycle. These glitches could be power, EM[114], or laser/optical [124]. We are assuming there is no external clock in this work, thus are ignoring clock perturbations as a valid glitch. This work will use power glitches as a demonstration of the usefulness of maintaining a phase-locked reference, such as is derived by the clock recovery scheme.

In addition a triggering mechanism that depends on waveforms in the analog data is demonstrated. How changes in operating frequency affect the triggering reliability is also explored, and it will be demonstrated that synchronous sampling provides a highly reliable data source for this trigger.

5.5.1 Sum of Absolute Difference Trigger

To inject a fault at a specific location, a pattern detection trigger called the Sum of Absolute Difference (SAD) is used. The implementation of the SAD comes from the ChipWhisperer system [105]. In this implementation 128 input samples, $\mathbf{T}$, are continuously compared to a 128 point reference waveform, $\mathbf{R}$, using (6.9). If the input was exactly the same as the reference waveform, the output of (6.9) would be 0. Normally the trigger condition is simply when the output of (6.9) falls below some

numerical value.

$$SAD = \sum_{p=0}^{127} |T_p - R_p| \tag{5.7}$$

If the data **T** has already been recorded (e.g. for resynchronizing recorded data), the form of (5.8) can be used. In this form an 'offset' parameter m is added, which slides the comparison window across all points in the recorded trace.

$$SAD(m) = \sum_{p=0}^{127} |T_{p+m} - R_p| \tag{5.8}$$

To determine the effect of varying clock frequency, a SAD reference waveform **R** will be compared to a recorded power trace **T**, where the same operation is occurring in both **T** and **R**. The frequency that the target is operating at when **T** is recorded varies, and the output of the SAD equation (5.8) is calculated. It is known a priori that when $m = 0$ the operations in both waveforms should be synchronized. Thus we would expect the following:

$$\arg\min_m (SAD(m)) = 0 \tag{5.9}$$

To determine the margin for the SAD trigger level, the minimum value of (5.8) is found when the offset is *not* zero, i.e. for all the *wrong* alignments of **R**. This is plotted against frequency in Fig. 5.22 — the distance between the two groups indicates the margin available. This uses a normal asynchronous capture, and note the SAD trigger would only function at a very narrow window around the reference trace waveform, which was captured when running at about 7.4 MHz.

By comparison, if we use synchronous sampling the SAD triggering is able to reliably detect the triggering point for **T** being recorded with a device frequency between 4.2 MHz – 13 MHz, even though the reference **R** was recorded at a different device operating frequency (about 7.6 MHz). At the extreme lower end of the operating frequency range the SAD triggering is not reliable, as around 3.9 MHz it would select the *wrong* triggering point.

For using the SAD triggering, hardware clock recovery is required if the device frequency is not constant. We will next consider not only the triggering of glitches, but the actual parameters defining the glitches as the device frequency varies.

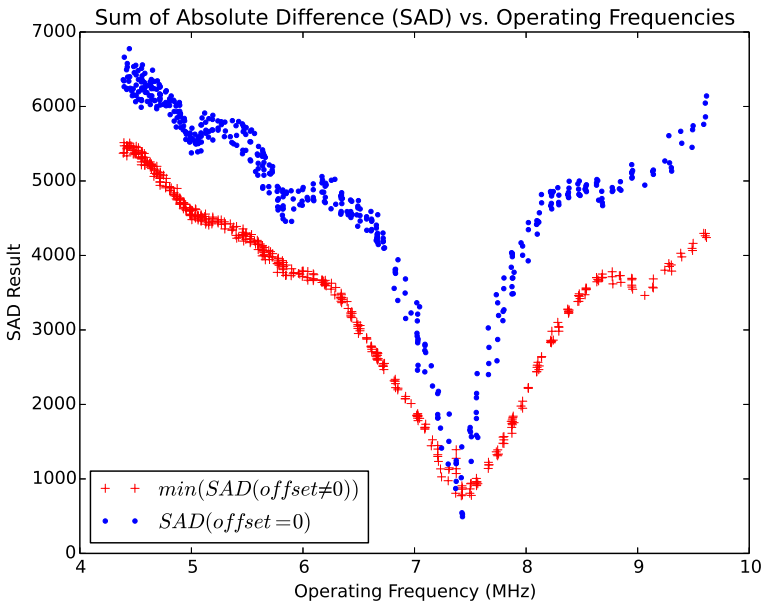

Figure 5.22: Output of (5.8) for the offset m being zero or non-zero. When the offset $m = 0$, this means the SAD output for the correct alignment of traces. When $m \neq 0$, this means the best possible SAD output for incorrect alignment of traces. Data sampled asynchronously at 312.5 MS/s.

5.5.2 Fault Injection and Target Code

For generation of faults, power glitching is used. A MOSFET is used across the power pins of the chip; the MOSFET forms a voltage divider with the shunt resistor being used for side channel power analysis measurement, and allows quickly dropping the voltage on the VCC pin. An example of the glitch waveform is shown in Fig. 5.24.

This setup allows power consumption to be monitored (required for the SAD trigger) along with monitoring the glitch status. The width and offset of the glitch is controlled via the ChipWhisperer system. The glitch width and offset is based on a percentage difference from the 'source clock'. If the device clock is known, this allows the width and offset to scale with changes in frequency, and ensures perfect synchronization of glitch location relative to clock edges. The ChipWhisperer system has high resolution on the glitch width and offset, having approximately 100 pS resolution on these options.

Where the source clock *isn't* known, i.e. without using clock recovery, an asynchronous clock is instead used to generate the glitch width and offset. In this case

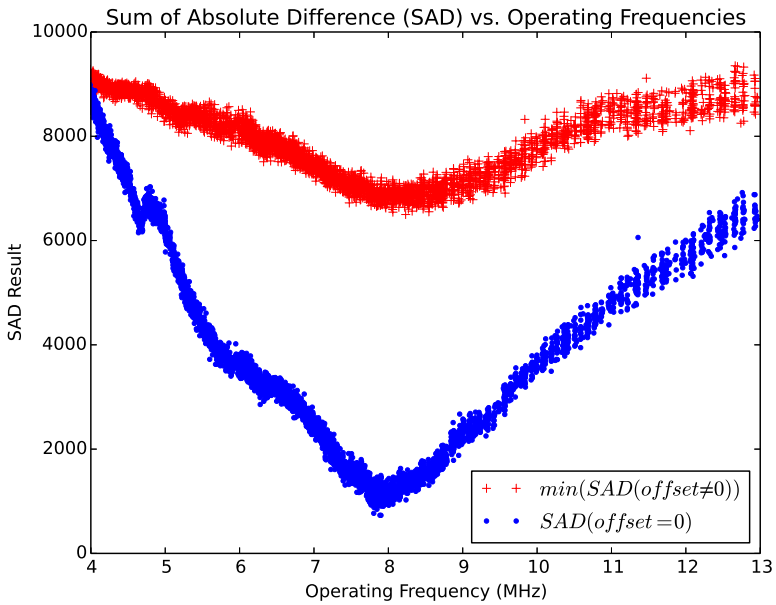

Figure 5.23: Output of (5.8) for the offset m being zero or non-zero. When the offset $m = 0$, this means the SAD output for the correct alignment of traces. When $m \neq 0$, this means the best possible SAD output for incorrect alignment of traces. Data sampled synchronously at $4\times$ device clock.

the glitch offset will occur relative to the trigger event, however the glitch parameters do not scale with device frequency, since the device frequency is not known.

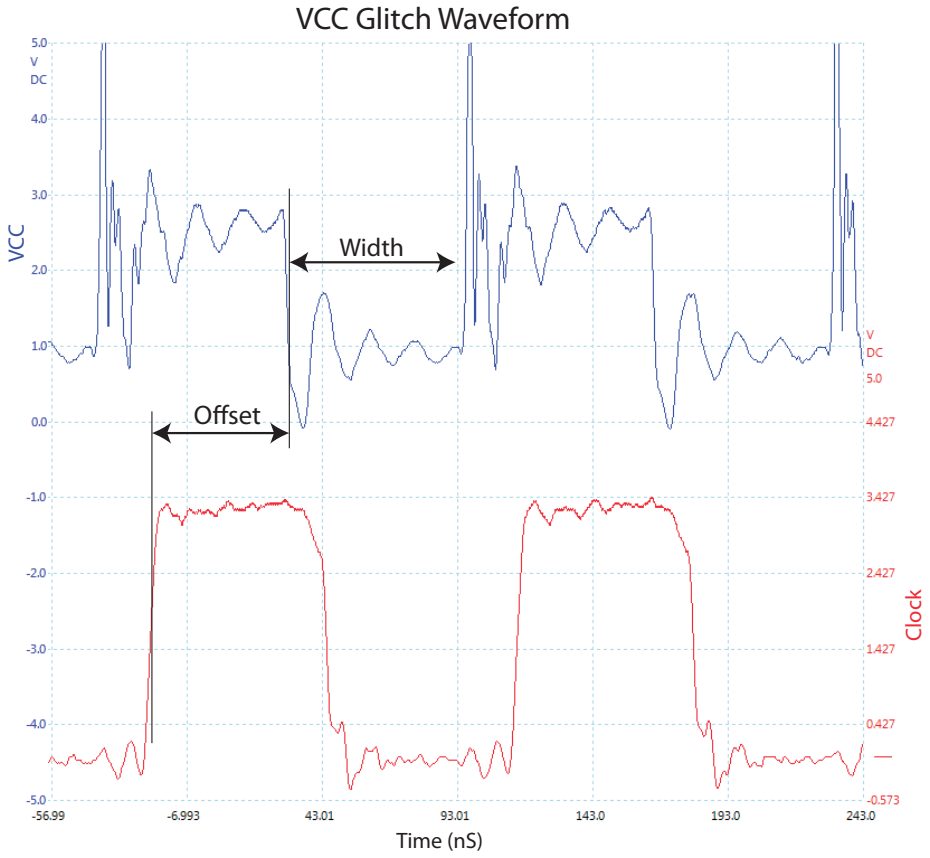

Figure 5.24: The VCC glitch inserted into the AtMega48A device for this test is derived from a source clock. The glitch width and offset are a function of that source clock, see [105] for details.

Listing 5.1: The source C code for the AtMega48A on which the glitch is tested.

```
#define OSCUART   94
#define OSCGLITCH 105

void glitch()
{
volatile uint8_t a = 0;

//Set frequency then TX
OSCCAL = OSCUART;
_delay_ms(10);
output_ch_0('A');

//Wait for character to TX
//then change frequency
_delay_ms(10);
OSCCAL = OSCGLITCH;
_delay_ms(10);

//Trigger Line
PORTC = 0x01;
PORTC = 0x00;

//Sensitive Loop
while(a != 2){
;
}

//Padding after loop
nop(); nop(); nop();
nop(); nop(); nop();

//Set frequency back
OSCCAL = OSCUART;
_delay_ms(10);
ch0_puts("1234");
}
```

5.5.3 Dependency on Target Frequency

As previously mentioned, the use of clock recovery is required for the SAD triggering to function. To allow comparison of glitch insertion with and without clock recovery, the AtMega48A is also programmed to set an IO line high at the moment where a glitch should be inserted. The glitch can thus be triggered even if the SAD trigger cannot be used, although in real systems it's unlikely such a trigger would exist. This trigger occurring at the moment of glitch insertion also means there is no error due to a differing number of device cycles between the trigger event and actual glitch, as would be the case if glitch insertion had a time-based offset from the trigger.

The code being glitched is shown in Listing 5.1, where a successful glitch is one which breaks out of the loop, without skipping past the padding. This allows a simple test to check if the glitch is causing the desired effect. A metric of the percent of glitches causing the desired effect that '1234' is printed is used to compare efficiency, which is averaged over 100 glitches.

The glitch offset and width is varied until what appears to be the maximum success rate is found. In one case the glitch width and offset scales with frequency (i.e. the device clock is fed into glitch generation), in the other the glitch width and offset is constant. The hardware is the same on both cases, again the AtMega48A device with an internal RC oscillator being used as the device clock.

It can be seen from the results of Fig. 5.25 that using the clock-synchronous glitch not only provides a more reliable glitch, but requires less tuning of parameters for operation over different frequencies in this example.

Considering that the synchronous capture provides the additional advantage of a useful SAD triggering system and the ability to easily count clock cycles from a trigger event, the clock recovery and synchronous capture method proposed here should have significant performance gains for fault injection.

5.6 Summary

It is know that *compression* of the power traces can be performed post-capture to reduce them to points of interest. Using synchronous sampling, however, eliminates the processing requirement, and makes triggering such as the Sum of Absolute Difference

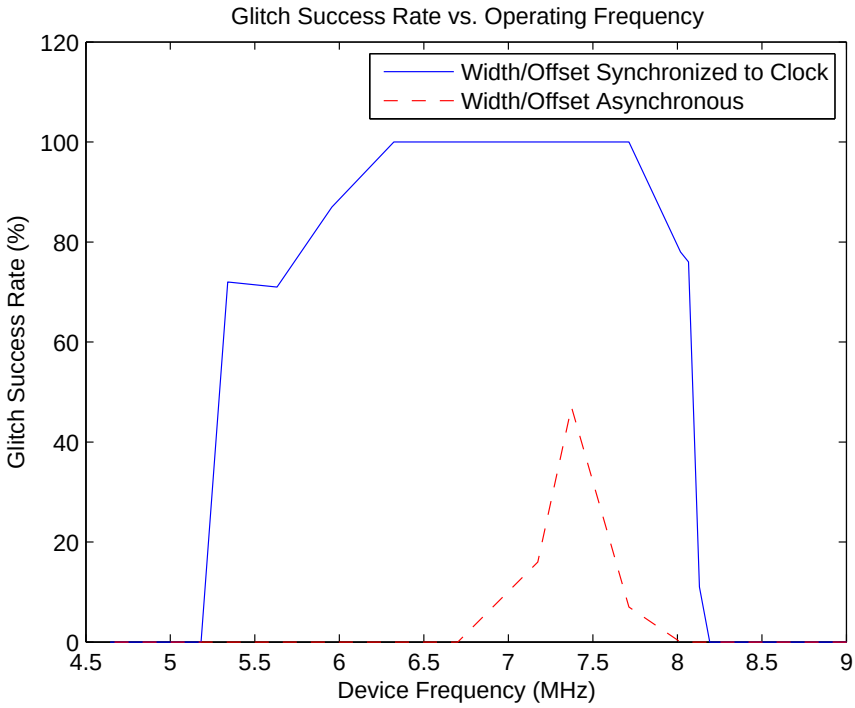

Figure 5.25: Success of voltage glitching where the glitch parameters are fixed to maximize success at a clock frequency of 7.37 MHz, and then device is then operated at different frequencies.

(SAD) mechanism reliable across operating frequency.

Synchronous sampling depends on the availability of the device clock, where many real devices contain an internal oscillator with no external signal. This chapter has demonstrated how a 'clock recovery' technique can generate an external reference clock which is phase-locked to the internal oscillator of the device.

If the device under attack is varying the internal oscillator, this external clock will remain phase-locked to the true frequency. As synchronous sampling is measuring clock edges and not absolute time, this varying clock has very little effect of the success rate of an attack performed on these traces. The traces remain well synchronized despite the changing clock frequency, with the exception of a phase offset due to delay in the filter.

This recovered clock is also useful for fault injection, where it is desired to insert a fault at some specific clock cycle or portion of a clock cycle.

In addition to hardware-based solutions, this chapter has also demonstrated the use of clock recovery with a standard asynchronous oscilloscope. This algorithm is of low complexity, and an implementation is available in the open-source ChipWhisperer project.

Chapter 6

AES-256 Bootloader Attack

This chapter is based on my paper previously published in [100].

6.1 Introduction

Details of side-channel power analysis have previously been presented in Chapter 2. This chapter will use the Correlation Power Analysis (CPA) attack by Brier et al. [22] against a bootloader with AES-256.

The chapter will first describe the bootloader in Section 6.2. A review of AES-256 will be given in Section 6.3, along with a discussion of side channel power attacks. There I will review the modifications required for attacking the full 32-bytes of the AES-256 key. Next in Section 6.4 I will briefly outline the hardware used in this chapter, and then describe the format of my results. Finally the results of a side-channel attack on the encryption key and initialization vector are described in Section 6.5 and 6.6.

Interested readers are referred to the open-source ChipWhisperer Project[1], where additional details including a step-by-step tutorial[2] of the attack are available.

6.2 Description of Bootloader

Rather than use a specific bootloader, a generic bootloader that can run on small microcontrollers will be presented. This bootloader in particular has been designed to closely follow the design from Atmel app-note AVR231 [7]. It can be appreciated that this bootloader is similar to other secure bootloaders available as application notes from vendors for other embedded microcontrollers, and this attack is not limited to Atmel's design.

[1] `http://www.chipwhisperer.com`

[2] See ChipWhisperer Documentation, available at `http://wiki.newae.com/Tutorial_A5_Breaking_AES-256_Bootloader`

A very simple encryption and communication protocol is used. The input data for the entire memory is split into 12-byte chunks, where the final block is padded with random characters. Each chunk has a 4-byte fixed signature sequence prepended which results in a 16-byte block.

The simple 4-byte signature is used to verify that any given 16-byte block was encrypted with the expected encryption key. As the bootloader is highly size-constrained, this simple signature verification is used over something such as a hash of the data.

Fig. 6.1 shows the generation of an encrypted block, along with the communications protocol. The communications protocol runs over a serial port, and contains a CRC-16 to verify that no communications errors occurred. The signature would also detect communications errors; but a signature failure is not communicated back to the sender to reduce the attack surface. Instead the CRC-16 is used so the sender can verify the correct data was sent, but the sender is supposed to be unaware if the bootloader is accepting the data (i.e. if the correct key was used).

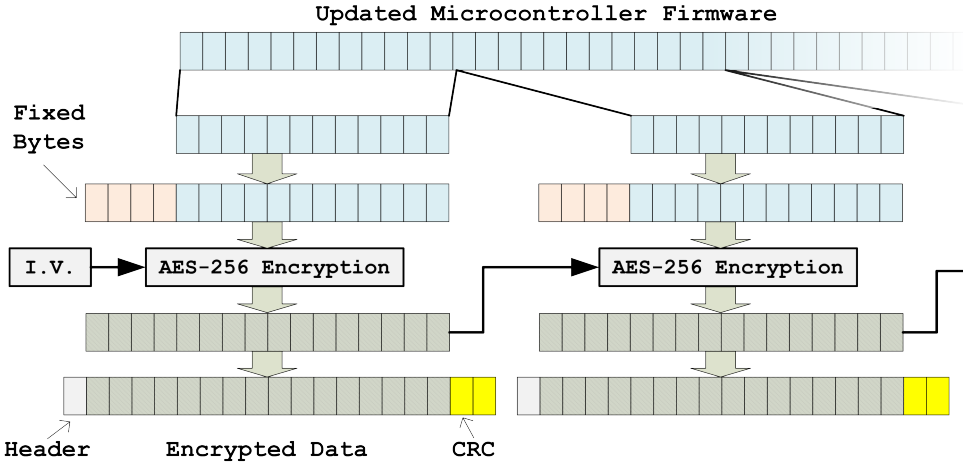

Figure 6.1: Data format for AES-256 bootloader showing both encryption format and communications protocol.

The implementation here does not actually write data to FLASH memory (i.e. it doesn't fully work as a bootloader), as this functionality is not required for this chapter. Details of the AES-256 decryption will be discussed next.

6.3 AES-256 Decryption

6.3.1 Background

The input cipher-text to the AES-256 decryption algorithm is C. The input key is 256 bits (32 bytes), which is expanded to 240 bytes, used 16 bytes at a time (each of the round keys). Each of the round keys is denoted as K^r, where the round $r = \{0, 1, 2, \cdots, 14\}$.

As AES decryption is performed with the same structure as AES encryption but 'in reverse', the first round of AES decryption in this work will be denoted as $r = 14$, the next round of AES decryption as $r = 13$, etc.

The input ciphertext C consists of 16 bytes:

$$C = [c_0, c_1, \cdots, c_{15}]$$

The AES algorithm stores an intermediate state X, which is updated after each round of the algorithm. The intermediate state for round r is denoted by a 16-byte array:

$$X^r = [x_0^r, x_1^r, \cdots, x_{15}^r]$$

The complete AES algorithm will use three special functions:

- Sub(): Performs bytewise substitution operation on X^r.
- Shift(): Shift Rows, reorders bytes in X^r.
- Mix(): Mix Columns, mixes bytes in X^r together.

All three functions have inverses, for example in the case of $SBytes()$ such that $\text{Sub}^{-1}(\text{Sub}(x)) = \text{Sub}(\text{Sub}^{-1}(x)) = x$.

With the Sub() and Shift() functions a single byte change in the input affects only a single byte of the output. With the Mix() function a single byte change in the input affects four of the output bytes.

The complete AES decryption algorithm can be described as in equations (6.1) to (6.5), where (6.3) is performed multiple times.

$$X^{14} = \text{Sub}^{-1}\left(\text{Shift}^{-1}(C \oplus K^{14})\right) \tag{6.1}$$

$$X^{13} = \text{Sub}^{-1}\left(\text{Mix}^{-1}\left(\text{Shift}^{-1}(X^{14} \oplus K^{13})\right)\right) \tag{6.2}$$

$$\cdots$$

$$X^{i} = \text{Sub}^{-1}\left(\text{Mix}^{-1}\left(\text{Shift}^{-1}(X^{i+1} \oplus K^{i})\right)\right) \tag{6.3}$$

$$\cdots$$

$$X^{1} = \text{Sub}^{-1}\left(\text{Mix}^{-1}\left(\text{Shift}^{-1}(X^{2} \oplus K^{1})\right)\right) \tag{6.4}$$

$$X^{0} = X^{1} \oplus K^{0} \tag{6.5}$$

6.3.2 Side Channel Power Analysis

When performing a side-channel power analysis attack, we will be attacking the value of X^{14}. We know the value of C (the input we sent the decryption algorithm), and perform a guess and check on each byte of K^{14}, where we use a Correlation Power Analysis (CPA)[22] attack with the Hamming Weight (HW) assumption on the leaked value of X^{14}.

The CPA attack requires us to attack each encryption subkey j (i.e. byte) of K^{14} independently. Using (6.1), we can calculate a hypothetical value of $X_j'^{14}$ based on the known ciphertext C_j, and some guess of the subkey value $K_j'^{14}$. If we see a large correlation between the hamming weight of our hypothetical value $X_j'^{14}$ and the power measurement trace related to the decryption of C, this suggests the guess of the subkey may be the correct value.

To determine the complete 32-byte encryption key, we will require both K^{14} and K^{13}. The classic CPA attack would only recover K^{14} as above, and a small change to the attack is required to find K^{13}.

Once K^{14} is known, the attack is re-run with the leakage function targeting X^{13}, where we wish to guess each byte of K^{13}. Due to the presence of $\text{Mix}^{-1}()$, it would appear that four bytes of the key must be guessed to achieve a single byte of X^{13}. This would entail guessing 2^{32} possibilities instead of 2^{8}, a considerably more challenging task.

We can however take advantage of the linearity of the $\text{Mix}^{-1}(a)$ operation to rearrange (6.2), as described in [98], and also demonstrated [93]. Using the property

that $\text{Mix}^{-1}(a \oplus b) = \text{Mix}^{-1}(a) \oplus \text{Mix}^{-1}(b)$, (6.2) becomes (6.6), where we are no longer guessing the key K^{13}, but a version of the key processed by $\text{Mix}^{-1}(\text{Shift}^{-1}(x))$:

$$X^{13} = \text{Sub}^{-1}\left(\text{Mix}^{-1}\left(\text{Shift}^{-1}(X^{14} \oplus K^{13})\right)\right) \tag{6.2}$$

$$X^{13} = \text{Sub}^{-1}\left(\text{Mix}^{-1}\left(\text{Shift}^{-1}(X^{14})\right) \oplus Y^{13}\right) \tag{6.6}$$

$$Y^{13} = \text{Mix}^{-1}\left(\text{Shift}^{-1}(K^{13})\right) \tag{6.7}$$

Once we fully determine Y^{13}, we can use (6.8) to determine the desired encryption key for round 13, K^{13}:

$$K^{13} = \text{Mix}(\text{Shift}(Y^{13})) \tag{6.8}$$

6.3.3 Cipher Block Chaining Mode

The bootloader uses AES-256 in Cipher Block Chaining (CBC) mode, where before being encrypted each block was XOR'd with the previous ciphertext. Since for the first block there is no previous ciphertext, a random Initialization Vector (IV) is used. The decryption flow is shown in Fig. 6.2, where the IV used for encryption must also be given to the bootloader. In this case the IV is programmed (along with encryption key) into the devices memory before deployment.

Note that if the decryption key is known, but the IV is not, this allows us to decrypt everything except the first block. If an attacker is simply looking to decrypt a file for reverse engineering purposes, they can probably derive enough useful detail without the first 16 bytes to accomplish this task.

6.4 Hardware

A bootloader as described in Section 6.2 is implemented in an Atmel AtMega328P-PU 8-bit microcontroller running at 7.37 MHz. Power measurements are taken using a resistive shunt inserted into the VCC line, where measurements are taken synchronously at 29.5 MS/s using a ChipWhisperer Capture Rev2 platform [105].

In Fig. 4.2 a photograph of the capture setup is shown. Details of the practicality of the attack will be discussed next.

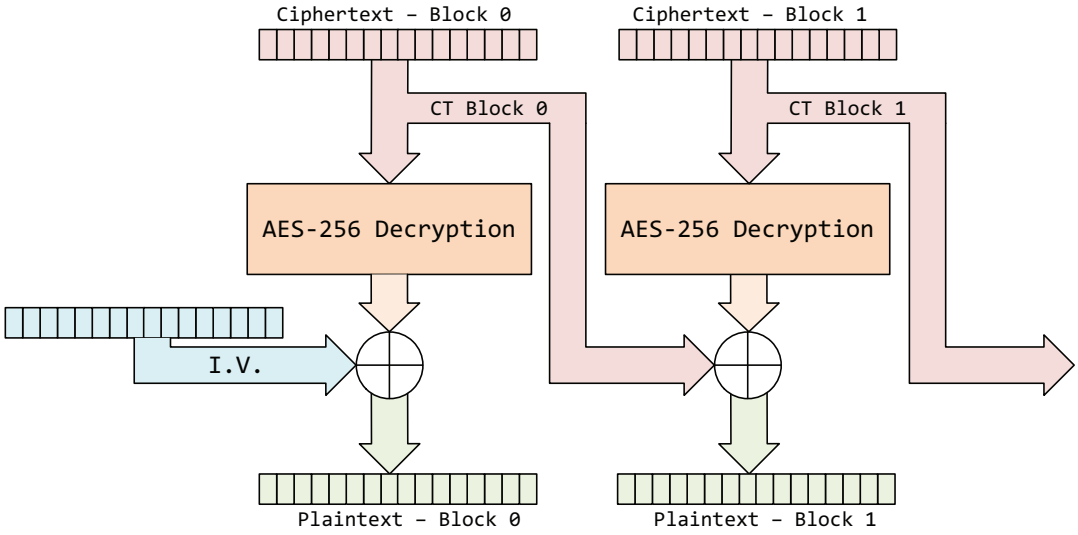

Figure 6.2: AES-256 Cipher Block Chaining (CBC) Mode Decryption

6.4.1 Triggering

Typical work demonstrating side-channel attacks uses an IO line of the microcontroller that indicates when the encryption (or decryption) routine is running. This provides an attacker with a perfectly synchronized trigger event, but in real implementations this will not be available.

For this work the Sum of Absolute Differences (SAD) trigger built into the ChipWhisperer is used. This allows triggering on a pattern in the analog waveform. The correct pattern can be determined through trial-and-error: it is known for example when the encrypted block was sent to the microcontroller, and we can infer that sometime after this event the decryption occurs. The SAD trigger can be used to 'walk through' the possible trigger events, until the analysis attack succeeds.

In this implementation of the SAD trigger 128 input samples, $\mathbf{T}$, are continuously compared to a 128 point reference waveform, $\mathbf{R}$, using (6.9). If the input was exactly the same as the reference waveform, the output of (6.9) would be 0. Normally the trigger condition is simply when the output of (6.9) falls below some numerical value.

$$SAD = \sum_{p=0}^{127} |T_p - R_p| \tag{6.9}$$

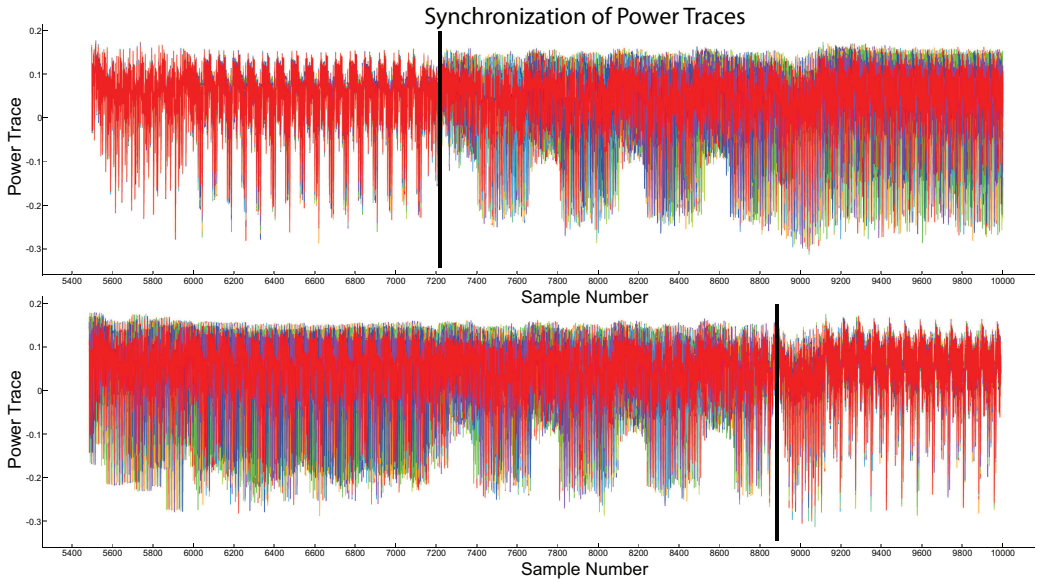

Figure 6.3: Power traces may not remain synchronized during the execution of the entire algorithm. The execution of the first round becomes unsynchronized around sample number 7300 in the top traces. They have been resynchronized in the lower example, allowing the attack to continue for the next round.

6.4.2 Synchronizing Traces

Traces may also need to be synchronized in time. In particular the AES-256 implementation used here has non-constant execution time, which does introduce another attack vector[73][74], but also means that the later rounds will not be perfectly synchronized even if the initial round is. This can be seen in the upper part of Fig. 6.3, where traces appear to become unsynchronized after a point in time.

To compensate for this a SAD resynchronization element is used during analysis for the 13^{th} round. In the lower part of Fig. 6.3 we can see traces appear synchronized toward the last half, but are now unsynchronized for the first half.

6.5 Determining Key

Details of the side channel analysis attack used are discussed in Section 6.3.2. The resulting GSR for the CPA attack on the 14^{th} and 13^{th} round encryption key is shown in Fig. 6.4.

The 14^{th} round key indicates the first 16 bytes recovered by the CPA attack. The 13^{th} round key is the next 16 bytes recovered, where we assume the first 16 bytes had

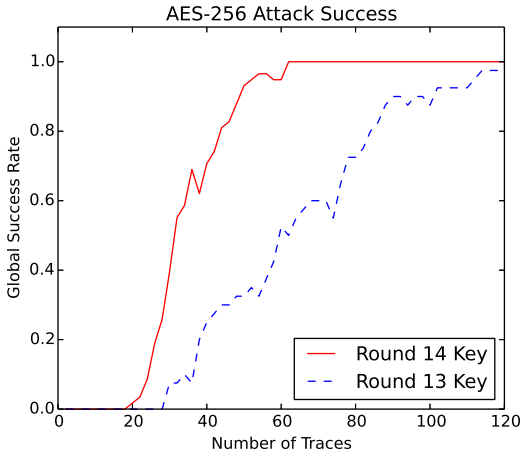

Figure 6.4: Two CPA attacks are performed to determine both the 14th and 13th round keys. Note the CPA attack on the 13th round key requires the 14th round key. The Global Success Rate(GSR) is displayed for the attack, where the attack has a very good chance of succeeding with 100 traces.

already been recovered. The 'total' success is given by the recovery of both the 14^{th} and 13^{th} round keys. With very good probability the entire encryption key can be recovered after 100 power trace measurements.

The PGE of the attack is given in Fig. 6.5. Again around 100 traces the PGE falls to zero indicating the key is perfectly known. Even with a smaller number of traces the guessing entropy is significantly reduced. The original PGE would be 128 for all subkeys, since each subkey is 8 bits, and we expect the correct key to be found half-way through, but with 60 traces this is reduced to an average PGE of only 2. This greatly reduced entropy could be attacked by brute-force guessing the most likely ranked keys.

6.6 Determining IV

If the plaintext was known, the Initialization Vector (IV) can be trivially determined once the encryption key is known. With the encryption key, the attacker can decrypt everything *except* the first 16 bytes; at this point they might be able to determine that the first 16 bytes of the plaintext were part of a fixed file header or similar material. In this case an attacker can determine the IV by XORing the expected plaintext with the output of the AES-256-ECB decryption function.

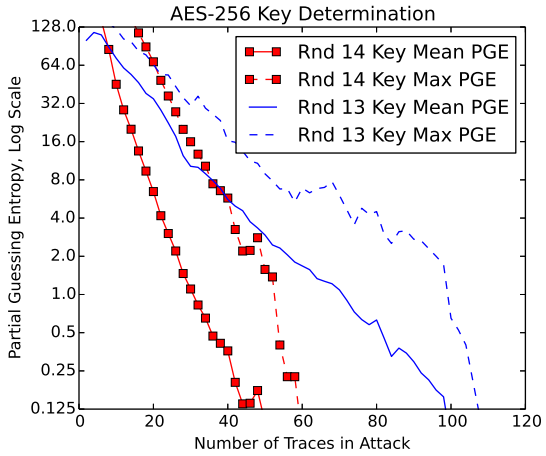

Figure 6.5: The Partial Guessing Entropy (PGE) data for the same attack in Fig. 6.4. The maximum PGE is the highest (worst) PGE of any of the 16 subkeys, and mean is the average of all subkeys for a given number of traces.

Without such knowledge, we could use a side-channel attack on the IV itself. We perform a CPA attack on the output of the decryption result XOR'd with the unknown IV, where we will guess each byte of the IV.

Fig. 6.6 shows the PGE for the IV. The CPA attack never fully recovers the IV even with 5000 traces, thus a GSR is not shown. We can consider the reason for this difficulty in obtaining a completely successful attack by reviewing again our leakage model and attack point. The application of the IV is as follows in (6.10).

$$P = X^0 \oplus IV \tag{6.10}$$

A single bit change in the IV will always result in a single bit change in the output. Thus it would be expected that guesses with small bit differences will be ranked similarly. When attacking the S-Box output in (6.1), a single-bit change in the input guess will result in multiple bits changing in the hypothetical output. Attacking the S-Box output means that wrong guesses have a considerably different hamming weight from incorrect guesses, and attack performance is considerably improved.

An example of the top-ranked guesses for byte seven of the IV is shown in Table I. In this case the PGE is two, as there are two wrong guesses for the IV byte ranked higher than the correct guess. Note the wrong guesses have a very close bit pattern to the correct value.

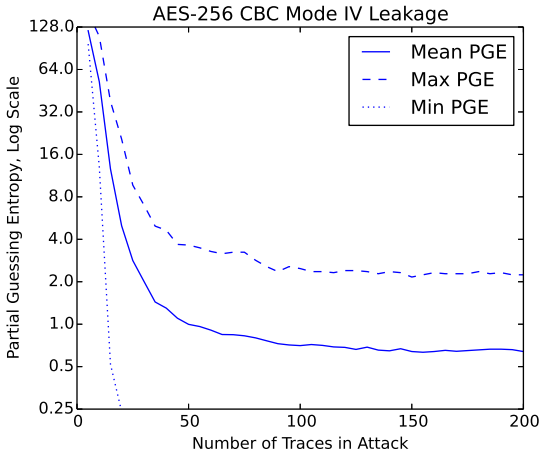

Figure 6.6: A CPA attack on the Initialization Vector (IV) being XORd with the plaintext shows that some guessing is still required, as the PGE never reaches 0 for all bytes. Instead an asymptotic behaviour is noted, which occurs due to several key hypothesis having the same correlation, as described in the text.

Table 6.1: Top three positive and negative correlation outputs for byte 7 of the I.V. Correct value of guess is `DA`.

Guess(Hex)	Guess(Bin)	Correlation
4A	01001010	0.8250
5A	01011010	0.8150
DA	11011010	0.7912
25	00100101	-0.7912
A5	10100101	-0.8150
B5	10110101	-0.8250

In addition, the absolute value of the correlation cannot be used. Due to the linear nature of (6.10), the correlation of the *bitwise inverse* of the correct guess would have the same absolute value as the correlation as the correct guess, but with the opposite sign. This is demonstrated in the lower three rows of Table 6.6. When attacking the S-Box we *can* use the absolute value of the correlation, since the S-Box is non-linear, and thus properties such as the bitwise inverse do not carry through the S-Box operation.

6.7 Determining Signature

It was also described in Section 6.2 that a secret 4-byte signature is added before each encrypted block. If the attacker wishes to have the bootloader accept a new data file, this signature must also be determined.

Provided the attacker has access to an encrypted firmware file, they can simply decrypt this file using the key determined with a CPA attack. The signature will be readily apparent due to the presence of a repeated fixed four-byte sequence in the decrypted file.

If attacking an 8-bit microcontroller, timing attacks are also possible on the signature check. If each byte of the signature is checked in sequence, it should be possible to determine from the power trace which byte failed on the signature check. This would require a partial brute-force attack, and is only relevant when the signature is checked byte-by-byte.

6.8 Summary

This chapter has explored a complete attack on a software implementation of AES-256-CBC used in a bootloader. This demonstrates the relevance of side-channel power analysis attacks to real systems, and not just academic implementations of the cryptographic algorithms.

Extending a standard CPA attack to work on AES-256 requires some modifications to the attack for the second decryption round, as detailed previously in [98] and [93]. In addition this chapter has demonstrated the use of a standard CPA attack to determine the Initialization Vector (IV), which in general demonstrates the effectiveness of a CPA attack on a single XOR operation. As many cryptographic algorithms use XOR, the results of the CPA attack on an XOR are of particular interest beyond just the attack on AES. The CPA attack on the XOR operation was part of the original CPA paper experiments[22], and this chapter provides some updated data for a recent 8-bit microcontroller.

Simply using a strong encryption such as AES-256 is insufficient to guarantee an embedded device will remain secure. A side-channel power analysis attack can be performed with a reasonable number of traces on a standard AES implementation,

revealing the encryption key. If protection against these attacks is required, countermeasures will need to be inserted into the AES implementation. The system designer must trade off the desired resistance to attacks against implementation complexity, and not simply assume that using a large key alone is sufficient to guarantee security.

Chapter 7

IEEE 802.15.4 Wireless Node Attacks

This chapter is based on my paper previously published in [104].

IEEE 802.15.4 is a low-power wireless standard which targets Internet of Things (IoT) or wireless sensor network (WSN) applications. Many protocols use IEEE 802.15.4 as a lower layer, including ZigBee (which encompasses many different protocols such as ZigBee IP and ZigBee Pro), WirelessHART, MiWi, ISA100.11a, 6LoWPAN, Nest Weave, JenNet, Thread, Atmel Lightweight Mesh, IEEE 802.15.5, and DigiMesh (this list only includes networking stacks that target commercial or industrial applications). As part of the IEEE 802.15.4 standard a security suite based on AES is included.

This chapter presents an attack against a wireless node that uses the IEEE 802.15.4 protocol. The following important results from developing this attack will be presented: (1) a shunt-based measurement method for devices with internal voltage regulators, (2) an attack against the hardware AES engine in the Atmel ATMega128RFA1, (3) an attack on AES-128 in CCM* mode as used in IEEE 802.15.4 [62], and (4) a method of causing the AES engine in the target device to perform the desired encryption. This attack is validated with a hardware environment (shown in Fig. 7.1).

The attack demonstrated here uses side-channel power analysis [72], specifically a correlation-based attack [22]. We obtained the power measurements in this work by physically capturing a node and inserting a shunt resistor. In general, side-channel attacks can be performed with a noncontact electromagnetic (EM) probe instead, which does not require modification to the device [50]. The EM measurement typically achieves similar results to the resistive shunt [2, 101].

This attack does not destroy the node under attack, and the node will continue to function during the attack. This makes detection more difficult: although a node is captured, it still appears on the network. The feasibility of capturing wireless nodes

and performing side-channel power analysis has previously been demonstrated against AES and ECC [38].

This previous demonstration was limited to software implementations of AES (i.e., not the actual hardware AES used by most nodes), and did not attack the AES-CCM* operating mode used by IEEE 802.15.4. Instead the attack in [38] assumed the encrypted data packet transmitted by the node allowed recovery of the last-round state of the AES algorithm. This is not the case in AES-CCM* used by IEEE 802.15.4 and most higher-layer protocols: recovering the last-round state would require a plaintext and ciphertext pair.

In practical scenarios the ability to capture a node, perform the attack, and return the node all within a short window reduces the risk of detection. The approach of [38] requires an attacker to passively wait for a transmissions to record power traces. While passively waiting is a reasonable approach for the 20–60 traces required by [38] to break a software AES implementation, this could entail an unreasonably long wait period for the thousands of traces typically required to break a hardware AES peripherals [71]. Our work allows an attacker to rapidly force the operation to occur, and collecting 20 000 traces can be accomplished in 15–60 minutes (depends on network stack and how much other traffic node must process).

This chapter results in a practical attack against IEEE 802.15.4 wireless nodes, the attack recovering the encryption key in use by the IEEE 802.15.4 layer. In addition the attack is demonstrated against a hardware AES peripheral as used by a standards-complaint IEEE 802.15.4 stack. This work is applicable to protocols that use IEEE 802.15.4 as a lower layer, even if these higher-layer protocols include additional security. The higher layer often uses the same vulnerable AES primitive as the IEEE 802.15.4 layer. Users of these protocols must carefully evaluate how the vulnerabilities detailed in this chapter might apply to the higher-layer protocols.

We begin by describing the attack on the ATMega128RFA1 AES hardware peripheral in Section 7.1. Next, we look at specifics of the use of AES encryption on the IEEE 802.15.4 wireless protocol in Section 7.2. This outlines the challenges of applying the side-channel attack to the AES-CCM* mode of operation, which is solved for the case of IEEE 802.15.4 in Section 7.3. Finally an application of this to a real IEEE 802.15.4 node is discussed in Section 7.4.

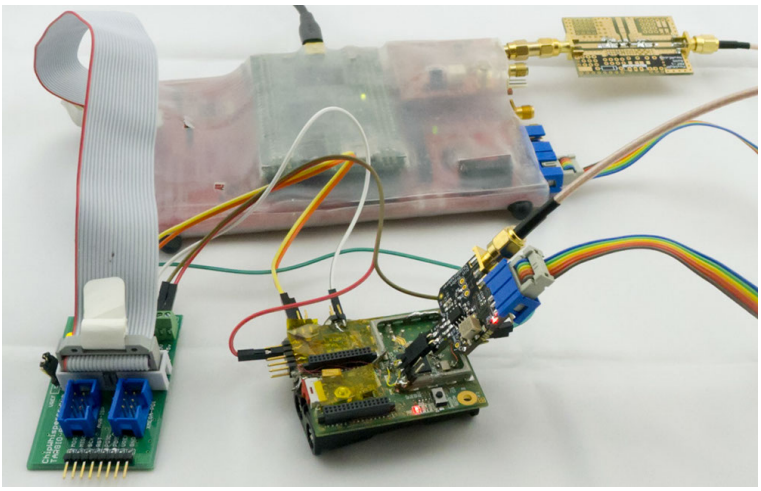

Figure 7.1: The ChipWhisperer capture hardware is used in this attack, although a regular oscilloscope will also work.

7.1 ATMega128RFA1 Attack

The Atmel ATMega128RFA1 is a low-power 8-bit microcontroller with an integrated IEEE 802.15.4 radio, designed as a single-chip solution for Internet of Things (IoT) or wireless sensor network (WSN) applications [9]. As part of the IEEE 802.15.4 radio module, a hardware AES-128 block is present, designed to work with the AES security specification of IEEE 802.15.4. Other examples of such chips (chips that include an IEEE 802.15.4 radio, microcontroller, and AES block) include devices from Freescale [48], Silicon Laboratories [122], STMicroelectronics [130], and Texas Instruments [134].

7.1.1 Side-Channel Power Analysis

Side channel power analysis was first reported in 1998 by Kocher et al [72], and was summarized in Chapter 2 of this thesis. Side-channel power analysis has previously been used to break a variety of hardware devices. Table 7.1 summarizes published power analysis attacks against commercially available hardware cryptographic devices. This table does not include software implementations running on commercially available hardware or hardware implementations that are not commercial products (i.e., research projects).

From Table 7.1, it can be seen that the CPA attack is an extremely popular attack

Table 7.1: Power analysis attacks against commercially available *hardware* cryptographic implementations. Entries marked with † indicate firmware-based implementations, but still being commercially available.

Target	Cipher	Attack	Ref.
CryptoMemory	proprietary	CPA	[11]
DESFire MF3ICD40	3DES	CPA	[109]
DS2432, DS28E01	SHA-1	CPA	[107]
Microchip HCSXXX	KeeLoq	CPA	[46]
ProASIC3	AES	PEA	[125]
SimonsVoss†	proprietary	CPA	[111]
Spartan-6	AES	CPA	[92]
Stratix II	AES	CPA	[94]
Stratix III	AES	CPA	[133]
Virtex-II	3DES	CPA	[90]
Virtex-4, Virtex-5	AES	CPA	[93]
XMEGA	AES	CPA	[71]
Yubikey 2†	AES	CPA	[110]
TI MSP430FR5	AES	CPA	[91]

for targeting real systems. The only non-CPA attack in [125] was a new technique called pipeline emission analysis (PEA), used to break the ProASIC3 device. While each of the remaining attacks used CPA, additional work may be needed before applying the CPA attack. For example in the attack on the DESFire [109], a preprocessing technique realigned the traces before applying CPA.

There are more advanced attacks that use a *template* of the device leakage [30]. Such a template does not use a leakage model based on assumptions, but instead the leakage model is based on measurements of the target device as it performs known encryptions. This requires that the attacker has access to a device that closely matches the target device that they can manipulate or program.

While such template attacks are considerably more powerful – being able to recover the encryption key with less measurements – the CPA attack using a simple assumption is more versatile, since it only requires access to the single target device of interest. A device vulnerable to a CPA attack will always be vulnerable to a template attack, and it is almost certain that the template attack will improve the success rate further. For this reason, this work deals solely with the CPA attack, which also aligns this work with previous publications of successful CPA attacks against commercially

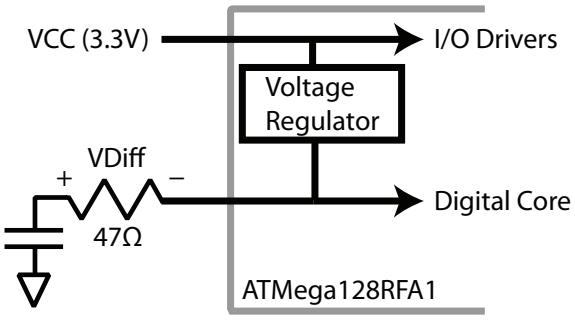

Figure 7.2: Because of the internal connection of the voltage regulator for the core voltage, the measurement shunt resistor must be mounted in the decoupling capacitor path.

available encryption hardware [46, 71, 90, 92, 93, 94, 109, 110, 107, 111, 133, 91].

This chapter uses the partial guessing entropy (PGE) to measure the attack success. PGE provides an indicator of the reduction in search space for each of the 16 key bytes. A PGE of zero for every key byte indicates that a full encryption key was recovered.

To perform this side-channel attack, I evaluate a method of physically measuring power on the ATMega128RFA1 in Section 7.1.2. I then determine an appropriate power model in Section 7.1.3, and I present the results of the CPA attack in Section 7.1.4. Additional considerations for attacking later rounds of the AES algorithm are presented in Section 7.1.5; these later-round attacks are required for the AES-CCM* attack.

7.1.2 Power Measurement

Power measurement is typically performed by inserting a resistive shunt into the power supply of the target device, and measuring the voltage drop across the shunt. Because devices often have multiple power supplies (such as VCC_{core}, VCC_{IO}, VCC_{RF}), the shunt must be inserted into the power supply powering the cryptographic core. As with many similar IEEE 802.15.4 chips [48, 122, 130, 134], the core voltage of the ATMega128RFA1 is lower (1.8 V) than the IO voltage (typically 2.8–3.3 V) [9]. Since these chips are designed to operate from a single 3 V coin cell battery, the lower core voltage reduces power consumption, whereas the higher IO voltage allows the device to operate directly from the coin cell.

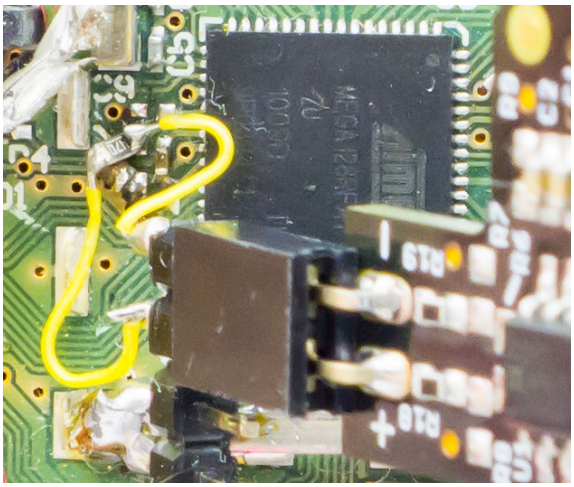

Figure 7.3: A 0603-sized 47-ohm resistor was inserted into the VCC_{core} decoupling capacitor, and a differential probe is used to measure across this resistor.

To avoid requiring an external voltage regulator for the lower core voltage, most of these devices also contain an integrated 1.8 V voltage regulator. Some devices require an external connection from the regulator output pin to the VCC_{core} pin. With this type of device we could perform the power measurements by either (a) inserting a shunt resistor between the output and input, or (b) using an external low-noise power supply with a shunt resistor (as in [38]). The ATMega128RFA1 is not such a device – it internally connects the regulator to the VCC_{core} pin, but does require a decoupling capacitor placed on the VCC_{core} pin (which also serves as the output capacitor for the voltage regulator). By inserting a shunt resistor into the path of the decoupling capacitor, we can measure high-frequency current flowing into the VCC_{core} pin. Note that this measurement will be fairly noisy, as we will also have noise from current flowing out of the voltage regulator. This is shown schematically in Fig. 7.2.

Fig. 7.3 shows the implementation of this arrangement, where a differential probe is placed across the resistor. An example of the power measurement resulting from this probe is shown in Fig. 7.4. A number of measurements with a regular oscilloscope are overlaid to provide an indication of the repeatability of the measurement.

7.1.3 Related Hardware Attack

The only previous published attack of an Atmel product with hardware AES acceleration was the XMEGA attack by Kizhvatov [71]. I used the XMEGA attack as a

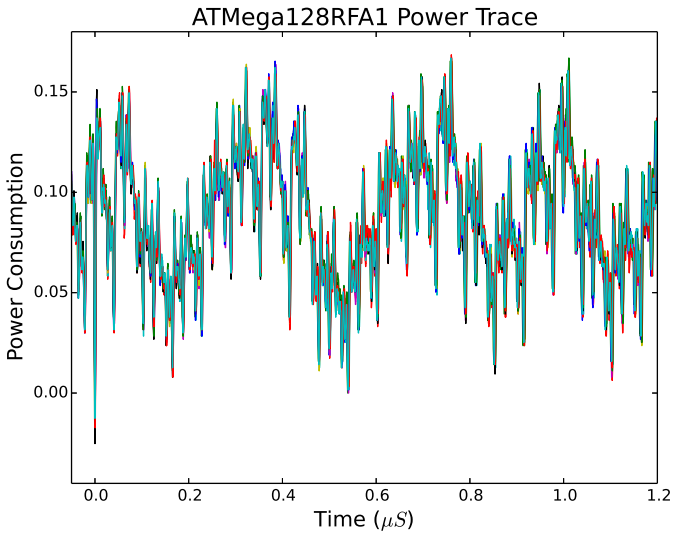

Figure 7.4: This figure shows the power trace for the first 1.2 μS of the AES-128 encryption. A total of ten such traces have been overlaid to demonstrate the consistent nature of the signal.

starting point, with the assumption that different Atmel products may use the same internal AES design.

Kizhvatov determined that for a CPA attack on the XMEGA device, a vulnerable sensitive value was the Hamming distance between successive S-box input values. These input values are the XOR of the plaintext with the secret key that occurs during the first `AddRoundKey`. This suggests a single S-box is implemented in hardware, with successive applications of the input values to the S-box.

The following notation considers P_j and K_j to be a byte of the plaintext and encryption key respectively, where $0 \leq j \leq 15$. To determine an unknown byte K_j, we first assume we know a priori the value of P_j, P_{j-1}, and K_{j-1}. The determination of K_{j-1} is presented later, but we can assume for now that byte K_{j-1} is known.

This allows us to perform a standard CPA attack, where the sensitive value is given by the Hamming weight of (7.1). That is to say the leakage for unknown encryption key byte j is: $l_j = HW(b_j)$. Provided K_0 is known, this attack can proceed as a standard CPA attack, with only 2^8 guesses required to determine each byte.

$$b_j = (P_{j-1} \oplus K_{j-1}) \oplus (P_j \oplus K_j), \quad 1 \leq j \leq 15 \tag{7.1}$$

As suggested in [71], if K_0 is unknown in practice, an attacker can simply proceed

with an attack for all 2^8 possibilities of K_0. The attacker may then test each of the resulting 256 candidate keys to determine the correct value of K_0. This would entail a total of $2^8 \times (2^8 \times 15)$ guesses.

For the specific case of K_0, a more straightforward approach exists. The author of [71] later determined that K_0 can be determined directly by using a leakage assumption based on the Hamming distance from the fixed value `0x00`. This leakage function is shown in (7.2).

$$l_0 = HW(b_0) = HW(P_0 \oplus K_0) \tag{7.2}$$

This allows the entire encryption key to be attacked with a total of 16×2^8 guesses.[1]

We now attempt to apply this attack to a different device, the ATMega128RFA1.

7.1.4 Application to ATMega128RFA1

The experimental platform was a Dresden Elektronik radio board, model number RCB128RFA1 V6.3.1. As mentioned previously, power measurements were taken by inserting a resistor between the VCC_{core} power pin and decoupling capacitor. A differential probe was used to measure the voltage across this resistor. Fig. 7.1 shows the complete capture setup.

To sample the power measurements, I used the open-source platform presented in Chapter 4. This capture hardware synchronizes its sampling clock to the device clock, and I configured it to sample at 64 MS/s (which is 4 times the ATMega128RFA1 clock frequency of 16 MHz).

To reduce noise in the power traces used for side-channel analysis, a band-pass filter with a passband of 3–14 MHz was inserted between the output of the differential probe and the low-noise amplifier input of the ChipWhisperer.

I implemented a simple test program in the ATMega128RFA1 that encrypts data received over the serial port on the experimental platform. This encryption is done via either a software AES-128 implementation or the hardware AES-128 peripheral in the ATMega128RFA1. When using the hardware peripheral, the encryption takes 25 μs to complete, or about 400 clock cycles.

[1] This is not published in their paper, but was described in private communication from the author.

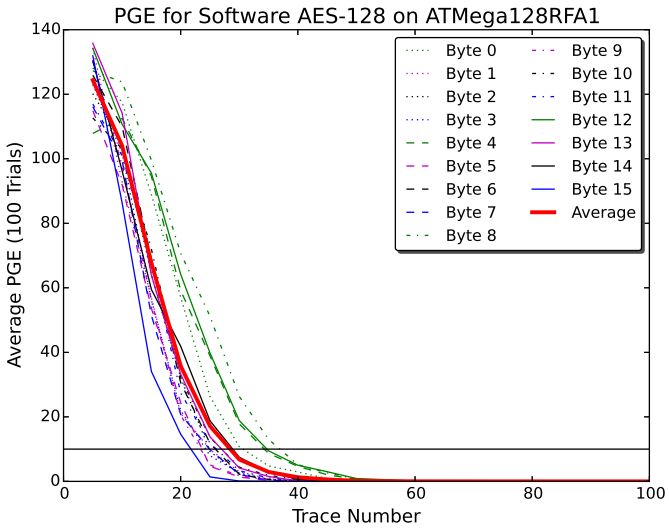

Figure 7.5: Attacking a software AES algorithm on the ATMega128RFA1 is used to confirm that the measurement setup is a viable method of measuring the leakage. Here the PGE across all bytes falls to zero in under 60 traces, completely recovering the key.

I used a standard correlation power analysis (CPA) attack [22], ranking the most likely byte as the one with the highest correlation values. To evaluate the measurement toolchain, I first performed an attack against a software AES implementation on the ATMega128RFA1.

Fig. 7.5 shows the results of the CPA attack against the software AES-128 implementation: I recovered the complete key in under 60 traces. These results can be compared to similar attacks using the ChipWhisperer hardware, where a software AES implementation on an AVR microcontroller is broken in around 30 traces [105].

I then recorded a total of 50 000 power traces, where the ATMega128RFA1 was performing AES-128 ECB encryptions using random input data during the time each power trace was recorded. For each trace, 600 data points were recorded at a sampling rate[2] of 64 MS/s. Each trace therefore covered about the first third of the AES encryption.

The initial CPA attack was repeated five times over groups of 10 000 traces. The resulting average partial guessing entropy for each byte is shown in Fig. 7.6. The first

[2]Note that this 64 MS/s sample rate is successful because the capture hardware samples synchronously with the device clock. If using a regular oscilloscope with an asynchronous timebase we expect a much higher sample rate to be required, similar to that reported in the XMEGA attack.

byte (which uses the leakage assumption of (7.2)) has the worst performance, as the guessing entropy does not reach zero with 10 000 traces.

Examples of the correlation output vs. sample point are shown in Fig. 7.7, which shows the peaks at the output of the correlation function on the CPA attack for the "correct" key guess. The sign of the peak is not important – the sign will flip depending on probe polarity – but note that the correct key guess results in a larger magnitude correlation than the incorrect guess at certain points. These points are when the physical hardware is performing the operation in (7.1).

Guessing of K_{j-1}

This attack used the leakage (7.2) of the first byte $j = 0$ to bootstrap the key recovery. Once we know this byte, we can use (7.1) to recover successive bytes.

Practically, we may have a situation where $j-1$ is not recoverable. Previous work assumed either some additional correlation peak allowing us to determine $j-1$, or the use of a brute-force search across all possibilities of the byte $j-1$ [71]. We can improve on this with a more efficient search algorithm, described next.

The leakage function (7.1) could be rewritten to show more clearly that the leaked value depends not on the byte values, but on the XOR between the two successive bytes, as in (7.3).

$$b_j = (K_{j-1} \oplus K_j) \oplus (P_{j-1} \oplus P_j), \quad 1 \leq j \leq 15 \tag{7.3}$$

The side-channel attack can be performed with the unknown byte K_{j-1} set to `0x00`, and the remaining bytes are recovered by the CPA attack described previously. These recovered bytes are not the correct value, but instead provide the value that has to be XOR'd with the previous byte to generate the correct byte.

The 256 candidate keys can then be generated with almost no computational work, by iterating through each possibility for the unknown byte K_{j-1}, and using the XOR values recovered from the CPA attack to generate the remaining byte values $K_j, k_{j+1}, \cdots, k_J$.

This assumes we are able to directly test those candidate keys to determine which is the correct value. As is described in the next section, we can instead use a CPA attack on the next-round key to determine the correct value of K_{j-1}.

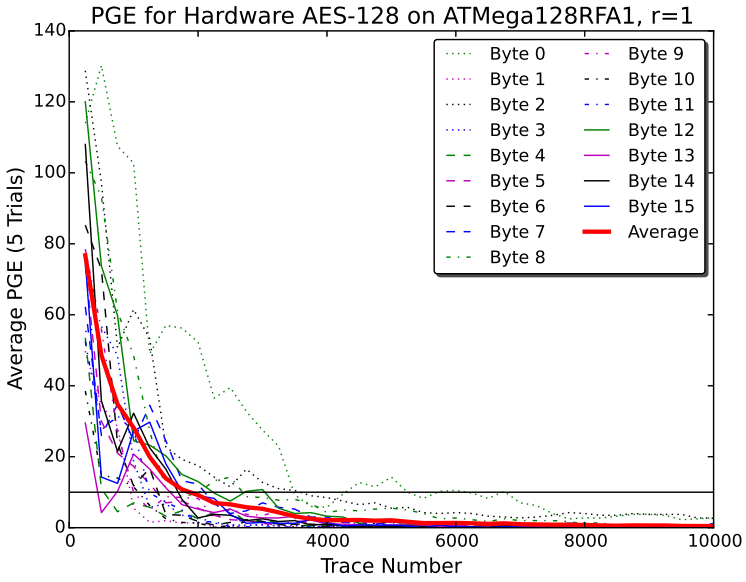

Figure 7.6: The CPA attack on the hardware AES peripheral reduces the guessing entropy to reasonable levels in under 5000 traces, and is able to recover the key in around 10 000 traces.

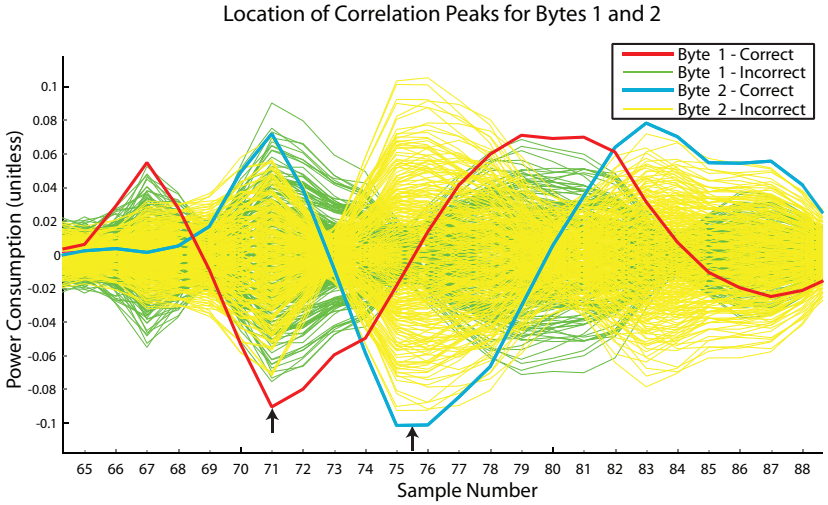

Figure 7.7: Correlation peaks for byte $j = 1$ and $j = 2$. The "incorrect guess" means the $2^8 - 1$ guesses which are not the value of K_j. The sample number refers to the sample points since start of the encryption operation, again sampling at 64 MS/s.

7.1.5 Later-Round Attacks

Whereas previous work has been concerned with determining the first-round encryption key, we will see in Section 7.3 that information on later-round keys is also required.

I determined that for later rounds the leakage assumption of (7.1) and (7.2) still holds, where the unknown byte K_j is a byte of the round key, and the known plaintext byte P_j is the output of the previous round. We can extend our notation such that the leakage from round r becomes $l_j^r = HW(b_j^r)$, where each byte of the round key is k_j^r, and the input data to that round is p_j^r.

Examples of the PGE when attacking the start of the third round ($r = 3$) are given in Fig. 7.8. The entropy change for all rounds tested ($r = 1, 2, 3, 4$) was similar.

For details of the execution time of the hardware AES implementation, refer to Table 7.2. This table shows the samples used for each byte in determining the most likely encryption key for the first four rounds. For byte 0 (the first byte), (7.2) is the sensitive operation. For later bytes (7.1) is the sensitive operation.

Note the sample rate is four times the device clock, and in Table 7.2 the sample delta from start to end of the sensitive operations within each round is about 64 samples, or 16 device clock cycles. This suggests that a sensitive operation is occurring on each clock cycle. Each round takes approximately 32–34 cycles based on the repeating nature of the leakages in later rounds.

Determining K_{j-1} Using Later Rounds

As described in Section 7.1.4, we can perform the CPA attack on byte K_j where K_{j-1} is unknown by determining not the value of the byte, but the XOR of each successive byte with the previous key. This means performing the attack first where K_{j-1} is assumed to be `0x00`.

By then enumerating all 2^8 possibilities for K_{j-1}, we can quickly generate 2^8 candidate keys to test. But if we are unable to test those keys, we need another way of validating the most likely value of K_{j-1}.

If we know the initial (first-round) key, we can determine the input to the second round, and thus perform a CPA attack on the second-round key. Instead we have 256 candidates for the first round ($r = 1$), and want to determine which of those keys is

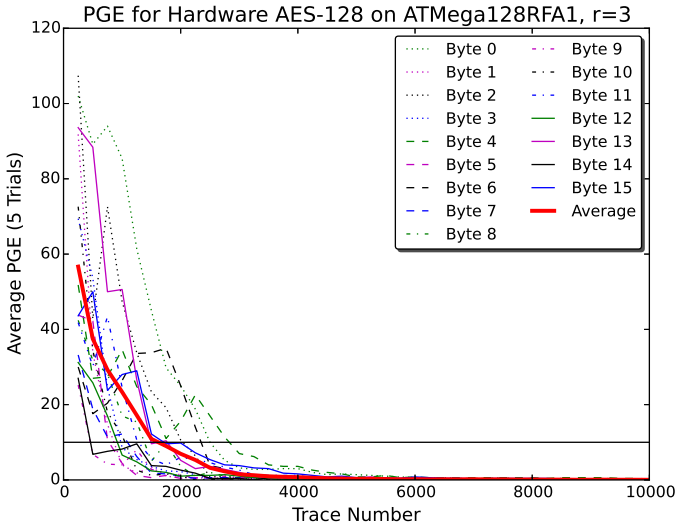

Figure 7.8: Attacking later rounds in the AES peripheral is also successful using the same leakage assumptions as the first-round attack.

Table 7.2: A small range of points is selected from each trace, corresponding to the location of the device performing (7.2) for $j = 0$, or (7.1) for $j \geq 1$. The variable r corresponds to the AES round being attacked, and j is the byte number.

j	$r = 1$	$r = 2$	$r = 3$	$r = 4$
0	66–70	198–204	336–342	474–478
1	70–75	205–210	340–345	478–481
2	73–78	208–215	345–348	482–489
3	79–83	213–216	350–355	486–490
4	81–88	218–221	355–368	490–494
5	85–90	220–225	358–361	494–498
6	89–95	225–233	362–365	498–501
7	93–98	230–235	366–370	502–505
8	98–102	233–237	370–374	506–508
9	101–106	237–241	373–377	510–513
10	106–111	240–247	378–383	514–519
11	110–114	245–250	382–385	518–521
12	114–119	248–254	385–390	522–524
13	118–123	253–258	390–394	525–529
14	121–126	258–265	394–398	530–534
15	126–129	262–268	398–402	534–538

correct.

To determine which of the keys is correct, we can perform a CPA attack on the first byte of the second round, K_0^2, repeating the CPA attack 256 times, once for each candidate first-round key.

The correlation output of the CPA attack will be low for all guesses of K_0^2 where $\mathbf{K^1}$ is wrong, and only for the correct guess of K_0^2 and $\mathbf{K^1}$ will there be a peak.

This technique will be used in Section 7.3.1, where we cannot test candidate keys as we are not recovering the complete key.

7.2 IEEE 802.15.4 Security

IEEE 802.15.4 is a low-power wireless standard, sending short data packets of up to 127 bytes at bit-rate of 250 kbit/s. Devices running IEEE 802.15.4 can achieve extremely low power consumption, running for years on a small battery [59]. When we refer to "IEEE 802.15.4," we are specifically targeting the IEEE 802.15.4-2006 standard.

The IEEE 802.15.4 standard is generally used as a lower layer with another network on top, as IEEE 802.15.4 does not specify details such as routing or distribution of keying material. Examples of popular higher-layer protocols include ZigBee and ISA100.11a. These higher-layer protocols often use IEEE 802.15.4 level security in combination with higher-level security – but this higher-layer security frequently uses the cryptographic primitives provided by IEEE 802.15.4. This chapter demonstrates the vulnerabilities of these primitives, and those vulnerabilities may also exist in higher-layer protocols.

The IEEE 802.15.4 standard uses AES-128 as the basic building block for both encryption and authentication of messages. The standard defines a mode of operation called CCM*, which extends the regular CCM mode by allowing the use of encryption without authentication [62].

CCM itself is a combination of counter mode of AES with cipher block chaining message authentication code (CBC-MAC) [145]. For the side-channel attack, we are only concerned with the details of data passed to the AES-128 block, and not the further processing that occurs after this block.

The AES-128 block itself is used in AES-CTR mode, with an input format as

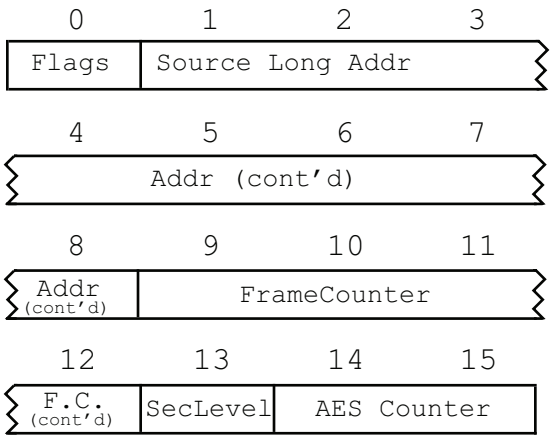

Figure 7.9: The following data is used as the input to AES-128 when a frame is decrypted by an IEEE 802.15.4 stack. The `FrameCounter` can be controlled by the attacker.

shown in Fig. 7.9. The first 14 bytes are the nonce, and the last two bytes are the AES-CTR mode counter. Each received frame must use a new nonce, as the counter itself only counts the number of 16-byte blocks in the frame.

To ensure that the nonce is fresh, a field called `FrameCounter` is included with each transmitted message and used as part of the nonce. The receiver verifies that the value of `FrameCounter` is larger than any previously used value, avoiding the reuse of a nonce.

A typical secure IEEE 802.15.4 wireless packet adds a message authentication code (MAC),[3] which ensures both the integrity and authenticity of this message. The MAC and payload can optionally be encrypted. Although the MAC is optional, configuring nodes to require a MAC is generally recommended, since accepting encrypted but unauthenticated packets presents a serious security risk [118].

The address and header information are never encrypted. This is mostly because it significantly simplifies message filtering: otherwise nodes would need to decrypt every message to determine the address information.

On receiving a packet, the IEEE 802.15.4 layer first returns an acknowledgment to the sender. If the packet has security enabled (it is encrypted or simply has

[3]The name message integrity code (MIC) is used in place of message authentication code (MAC) within the IEEE 802.15.4 standard, as the acronym MAC already refers to the medium access control layer. This chapter uses MAC to mean message authentication code, and medium access control is spelled out when required.

a MAC appended) the following operations are performed on the received packet, where processing stops if a step fails:

1. Validate headers and security options.

2. Check that the received frame counter is numerically greater than the last stored frame count.

3. Look up the secret key based on message address and/or key index.

4. Decrypt the payload (and MAC if present).

5. Validate the MAC (if present).

6. Store the frame counter.

For our side-channel attack we only care that step 4 is performed; this means our packet must successfully pass through steps 1–3. This requires that the packet is properly addressed and has an acceptable security configuration, such as using a valid key identifier and address. Generating such a message is discussed next.

7.2.1 Detailed Message Format

As mentioned, we need to ensure our 802.15.4 message is decrypted by the target device. Specific requirements vary depending on the network configuration, but as an example the frame used in our experiments was:

```
09 d8 01 ff ff ff ff ba ad 01 02 03 04 05 06 07 08 0d FC FC FC FC 01 AA
AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA 00 00
```

Where details of each portion of the frame are described as follows:

`09 d8`	Frame header
`01`	Sequence number
`ff ff ff ff`	Broadcast frame
`ba ad`	Source network ID
`01 02 ···`	Source address
`08`	
`0d`	Security level
`FC FC FC FC`	`FrameCounter` used as part of the AES nonce
`01`	Key ID
`AA AA ···`	16 bytes of encrypted
`AA`	data and 4 bytes of MAC
`00 00`	Replace with CRC-16

The encrypted data and MAC are not used in the attack; only the value of the `FrameCounter` is used. The attacker can send a frame with random values inserted into the `FrameCounter`; they only need to ensure the random values are larger than the last *valid* received value.

Since the attacker's packets will be rejected as invalid once decrypted, the attacker will *not* update the internal frame counter field. An attacker could discover the approximate current `FrameCounter` value through sniffing valid packets sent to the device, since the frame counter is sent unencrypted.

Alternatively, an attacker can simply send very high values – the attacker could ensure the highest two bits are always `10`, `01`, or `11`. Provided the node has received fewer than a billion messages using the same key, this frame counter will be accepted.

The example message used here is sent as a broadcast frame, which ensures it will be received and processed by the target node. Using a broadcast message also hides the target of the attack, and someone sniffing the airwaves might simply assume that a device on a separate network is sending encrypted messages. They might assume either this node is malfunctioning, or the node is using some proprietary protocol that sends encrypted broadcast messages.[4]

[4]Sending a data message that is broadcast across both the network ID and device address is

7.2.2 Brute-Force Search

In Section 7.1.1, the side-channel attack is described as not always fully recovering the encryption key. This necessitates a key search, which requires a comparison function such as having a plaintext/ciphertext pair.

For IEEE 802.15.4, any message with a MAC present can be used as this plaintext/ciphertext pair. It is sufficient for the attacker to sniff a single wireless message secured with a MAC using the target key; the attacker can then use a given hypothetical key to calculate the hypothetical MAC, and compare the resulting MACs to determine if the hypothetical key matches the true key.

This attack is trivial in practice: most networks using 802.15.4 security have message authentication enabled (i.e., messages have MACs) [118], so the attacker can simply capture a packet directed at or sent from the target. As the MAC may be shorter than the key, they may wish to capture several packets to confirm the true key was found, and not a collision for this particular message.

7.3 Application to AES-CCM* Mode

For a standard CPA attack, we require the ability to cause a number of encryption operations to occur with known plaintext or ciphertext material. In addition, the data being encrypted must vary between operations, as otherwise each trace will generate the same hypothetical intermediate values during the search operation of the CPA attack.

From Section 7.2 and Fig. 7.9, we know that a number of the bytes are fixed during the AES encryption operation. Practically *all* the bytes except for the `FrameCounter` are considered fixed in this attack. The `Flags` and `SecLevel` bytes will have constant (and known) values. Initially it would appear that the `Source Long Address` and `AES Counter` fields may vary, but as we discuss next, this is not the case.

The `Source Long Address` field comes from internal tables in the 802.15.4 stack containing keying material, and is not simply copied blindly from the incoming packet address. This field can be considered effectively fixed. The `AES Counter` field changes

unusual. However, it would work, and it is not unusual for devices to be deployed in the field after only basic testing, leaving in place bad practices such as flooding the network with encrypted messages.

during operation, as it increases for each 16-byte block encrypted in AES-CCM* mode. But as the IEEE 802.15.4 packet is limited to a total of 128 bytes, the `AES Counter` field could never exceed `0x0007`. Thus, between these 10 bytes, at most 3 bits vary during operation.

We instead rely on the ability of the attacker to control the `FrameCounter` field to mount a successful attack on an IEEE 802.15.4 wireless node.

We will assume an attack on the first encryption operation when a packet is received, i.e. we do not take advantage of the fact that each received packet causes more than one encryption operation. Practically this means that when our work refers to requiring N encryption traces, we only need to send a smaller number (in the range $N/4$) packets to the wireless node.

7.3.1 Previous AES-CTR Attacks

The AES-CCM* mode used by IEEE 802.15.4 is a combination of CBC-MAC and CTR modes of operation. The attack is on the AES-CTR portion of the algorithm, with some modifications to reflect the use of a frame counter for the nonce material.

Previous work on AES-CTR mode has focused on the assumption that we can cause a number of encryptions to occur in sequence (i.e., with increasing counter number), but with unknown but constant nonce material [64]. This work uses many of the constructs developed by Jaffe in [64], but with different assumptions of inputs on the AES block and a different leakage model. These differences necessitate the development of new techniques to recover partial keying information, and we cannot simply apply the previous attack directly.

In this case, we have the ability to change 4 bytes of the input plaintext (bytes 9, 10, 11, and 12). The CPA attack only allows us to recover these four bytes of the key, so we push the attack into later AES rounds to recover the entire encryption key.

Initially, we can assume that the keying material associated with bytes 9–12 can be recovered by a standard CPA attack, as was shown in Section 7.1. The remaining bytes cannot be recovered, as the input data is constant.

Consider the steps in the AES-ECB encryption algorithm: $AddRoundKey()$, $SubBytes()$, $ShiftRows()$, and $MixColumns()$. All four functions can be assumed to operate on a 16-byte AES state matrix. The first three operate in a byte-wise

manner – that is, a single-byte change of the state matrix before the operation results in a single-byte change after the operation. The $MixColumns()$ operation introduces diffusion, which operates on a column of the AES state – a single-byte change in the AES state results in 4 bytes changing after the $MixColumns()$ operation.

For the $MixColumns()$ operation, we can represent the four input bytes – one column of the state matrix – with $S_0, \cdots, S_3$, and the resulting output bytes with $S'_0, \cdots, S'_3$. The $MixColumns()$ operation uses multiplication over the Galois field GF(2^8), where we represent this multiplication operation with the symbol "$\circ$". The $MixColumns()$ operation then becomes:

$$S'_0 = (2 \circ S_0) \oplus (3 \circ S_1) \oplus S_2 \oplus S_3 \tag{7.4}$$

$$S'_1 = S_0 \oplus (2 \circ S_1) \oplus (3 \circ S_2) \oplus S_3 \tag{7.5}$$

$$S'_2 = S_0 \oplus S_1 \oplus (2 \circ S_2) \oplus (3 \circ S_3) \tag{7.6}$$

$$S'_3 = (3 \circ S_0) \oplus S_1 \oplus S_2 \oplus (2 \circ S_3) \tag{7.7}$$

Where the input state to the $MixColumns()$ operation was the AES state after the $ShiftRows()$ operation. The output of the $MixColumns()$ operation is used as the input state for the next round.

Constant Inputs

For the $MixColumns()$ operation, we can lump all fixed input bytes into a single constant. As described in [64], if bytes S_1, S_2, S_3 of the input were fixed, we could rewrite (7.4)–(7.7) as:

$$S'_0 = (2 \circ S_0) \oplus E_0 \tag{7.8}$$

$$S'_1 = S_0 \oplus E_1 \tag{7.9}$$

$$S'_2 = S_0 \oplus E_2 \tag{7.10}$$

$$S'_3 = (3 \circ S_0) \oplus E_3 \tag{7.11}$$

Note that these bytes are constant but unknown. This occurs because although we do know the fixed plaintext input bytes (i.e., the nonce), we do not know the

keying material used for those bytes. These outputs will become the inputs to the next round of the AES algorithm.

Again using the method from [64], our objective is now to recover the next-round key. Consider that we have gone through one round of AES, and our objective is to recover the second-round key. We do not know the actual input data for this round, which is the output of the $MixColumns()$ step from the previous round. For example, to recover the first key byte K_0, we would need to know the output S_0 from $MixColumns()$. If some of the input bytes to $MixColumns()$ are fixed but unknown, we would instead recover the modified output S'_0.

Performing the CPA attack, we could recover instead a version of this key (we will refer to it as K'_0) XOR'd with the unknown constant E_0, that is $K'_0 = K_0 \oplus E_0$. We can use this modified key as one input to the $AddRoundKey()$ function, where the other input is our modified input to this round S'_0. Note that the output of $AddRoundKey()$ will be equivalent to the case where we had both the true key and true input:

$$\begin{aligned} AddRoundKey(K'_0, S'_0) &= K'_0 \oplus S'_0 \\ &= (K_0 \oplus E_0) \oplus (S_0 \oplus E_0) \\ &= K_0 \oplus S_0 \end{aligned}$$

This is sufficient information to perform the attack on the next round of the AES algorithm. If the entire modified version of a key can be recovered for a given encryption round, we can recover the entire *unmodified* key by attacking the next encryption round. This unmodified key can then be rolled backwards using the AES key schedule.

This fundamental idea is used to attack AES-CCM* as used in IEEE 802.15.4, where many of the input bytes are fixed. By pushing the attack into later rounds, we can recover fixed bytes with a regular CPA attack.

Description of Attack

We describe the attack by working through a symbolic example, using the following variables:

P_i^r : "text" input to the $AddRoundKey()$

K_i^r : "round key" input to the $AddRoundKey()$

E_i^r : a constant, see Section 7.3.1

n_i^r : the modified round key, $K_i^r \oplus E_i^r$

s_i^r : the output of the $SubBytes()$ function

v_i^r : the output of the $ShiftRows()$ function

m_i^r : the output of the $MixColumns()$ function

X : variable and known input plaintext values

Y : variable and known intermediate values

Z : variable and known intermediate values

N : known modified round-key values (n_i^r)

K : known key or round-key values (k_i^r)

c : constant values (may be known or unknown)

? : variable and unknown values

X* : group of variables which has a small set of possible candidates for the correct value

Initially, we have the known input plaintext, where 12 of the bytes are constant, and the 4 variable bytes are under attacker control (`FrameCounter`):

$\mathbf{p^1}$ `= [c c c c c c c c c c X X X X c c c]`

From this, we can perform a CPA attack to recover 4 bytes of the key. Note that in practice the byte K_9^1 cannot be recovered because K_8^1 is unknown. Instead we use the technique detailed in Section 7.1.5 to generate 256 candidate keys for $K_9^1, \cdots, K_{12}^1$, and test them at a later step. This means we can assume the following is the state of our initial-round key:

$\mathbf{r^1}$ `= [c c c c c c c c c c K*K*K*K* c c c]`

This can be used to calculate the output of the $SubBytes()$ and $ShiftRows()$ functions, where the majority of bytes are constant (but unknown):

$\mathbf{s^1}$ = [c c c c c c c c c Y*Y*Y*Y* c c c]

$\mathbf{v^1}$ = [c c Y* c c Y* c c c c c c Y* c c Y*]

At this point we need to symbolically deal with the $MixColumns(\mathbf{v^1})$ output, as we will be working with the modified output that has been XOR'd with the constant E described in Section 7.3.1. As in [64], this is accomplished in practice by setting unknown constants c to zero, and calculating the output of the $MixColumns(\mathbf{v^1})$ function. The unknown constants are all pushed into the variable E, which we never need to determine the true value of. This means our output of round $r = 1$ becomes:

$\mathbf{m^1}$ = [Z*Z*Z*Z*Z*Z*Z*Z* c c c c Z*Z*Z*Z*]

Note that 4 bytes of this output are constant. We again set these constant bytes to zero to simplify our further manipulation of them. This means our input to the next round becomes:

$\mathbf{p^2}$ = [Z*Z*Z*Z*Z*Z*Z*Z* 0 0 0 0 Z*Z*Z*Z*]

We are not able to recover $n_8^2, \cdots, n_{11}^2$ yet, as the inputs associated with those key bytes are constant.

We first attempt to recover n_0^2, which is performed for all 256 candidates for $K_9^1, \cdots, K_{12}^1$. As mentioned in Section 7.1.5, the highest correlation peak determines both $K_9^1, \cdots, K_{12}^1$ and n_0^2. This means we no longer have a group of candidates for the input, but a single value:

$\mathbf{p^2}$ = [Z Z Z Z Z Z Z Z 0 0 0 0 Z Z Z Z]

We can then proceed with the CPA attack on the remaining bytes of $\mathbf{n^2}$. Bytes $n_1^2, \cdots, n_6^2$ can be recovered by application of the CPA attack from Section 7.1.4.

Recovery of n_7^2 using the same process is not possible, as $MixColumns(\mathbf{v^1})$ interacts with the leakage model. The inputs to this round P_6^2 and P_7^2, which are generated by (7.6) and (7.7) respectively as the previous-round $MixColumns(\mathbf{v^1})$ outputs m_6^1

and m_7^1. When attacking n_7^2, we apply (7.1) to (7.6) and (7.7). This means our leakage is:

$$HW\left((n_6^2 \oplus (7.6)) \oplus (n_7^2 \oplus (7.7))\right) \tag{7.12}$$

The XOR cancels common terms in (7.6) and (7.7), and in this case that term is S_1. Here S_1 is the variable and known input to the $MixColumns(\mathbf{v^1})$, the result being that the leakage appears constant and the attack fails.

Instead, we can recover this value using a CPA attack on the next round, which is described later.

Returning to our CPA attack on the modified round key, we are unable to recover $n_8^2, \cdots, n_{11}^2$ as the associated inputs are constant.

As n_{11}^2 is unknown, we cannot directly recover $n_{12}^2, \cdots, n_{15}^2$. Instead we again use the method of Section 7.1.5 to generate 256 candidates for $n_{12}^2, \cdots, n_{15}^2$.

At this point we assume the CPA attack has succeeded, meaning we have recovered the following bytes of the *modified* round key, where the final 4 bytes are partially known – we have 256 candidates for this group, as we know the relationship between each byte, but simply don't know the starting byte to define the group:

$\mathbf{n^2}$ = [N N N N N N N c c c c c N*N*N*N*]

Remember, once we apply $AddRoundKey(\mathbf{n^2}, \mathbf{p^2})$, the constant E will be removed – E is included in both the output of $MixColumns(\mathbf{v^1})$ and the modified key – meaning we can determine the true value of the input to this round.

The outputs $8, \cdots, 11$ of $MixColumns(\mathbf{v^1})$ from the first round are constant, so we also know these inputs are constant, and the four unknown modified bytes $n_8^2, \cdots, n_{11}^2$ can be ignored at this point. The result of $AddRoundKey(\mathbf{n^2}, \mathbf{p^2})$ for these bytes will be another constant.

The unknown byte n_7^2 is associated with variable input data, meaning this output will be unknown and variable. At this point we can represent the known outputs of $SubBytes()$ and $ShiftRows()$:

$\mathbf{s^2}$ = [Y Y Y Y Y Y Y ? c c c c Y* Y*Y*Y*]

$\mathbf{v^2}$ = [Y Y c Y* Y c Y* Y c Y* Y ? Y* Y Y c]

As before, we can set unknown constant values to zero to determine the modified output $\mathbf{m^2} = MixColumns(\mathbf{v^2})$. The unknown variable byte means 4 bytes of the $MixColumns(\mathbf{v^2})$ output are currently unknown. In addition, we have 256 candidates for the remaining known values, since the four modified bytes $n_{12}^2, \cdots, n_{15}^2$ have been mixed into all output bytes by $ShiftRows(\mathbf{p^2})$ and $MixColumns(\mathbf{v^2})$:

$\mathbf{m^2}$ = [Z*Z*Z*Z*Z*Z*Z*Z* ? ? ? ? Z*Z*Z*Z]

This becomes the input to the next round:

$\mathbf{p^3}$ = [Z*Z*Z*Z*Z*Z*Z*Z* ? ? ? ? Z*Z*Z*Z]

We again apply the CPA attack on n_0^3 across all values for n_0^3 and the 256 candidates for the previous modified round key (a total of 2^{16} guesses), the peak telling us the value of n_0^3 and $n_{12}^2, \cdots, n_{15}^2$. We now know which of the candidates to select for further processing:

$\mathbf{p^3}$ = [Z Z Z Z Z Z Z Z ? ? ? ? Z Z Z Z]

We can apply a CPA attack to discover the modified key values $n_1^3, \cdots, n_7^3$. The unknown plaintext byte ? represents a changing value. We cannot ignore it as we can constant values in the $MixColumns(\mathbf{v^2})$, and thus cannot apply the CPA attack on the remaining bytes.

Instead we enumerate all possibilities for n_7^2, and apply a CPA attack against n_8^3, similarly to previously described attacks from Section 7.1.5. We verified experimentally that the correlation value with the highest peak for n_8^3 resulted only when n_7^2 was the correct value, as in Fig. 7.10. This means we now have the entire modified output of $MixColumns(\mathbf{v^2})$, and thus the complete modified input plaintext to round 3:

$\mathbf{p^3}$ = [Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z]

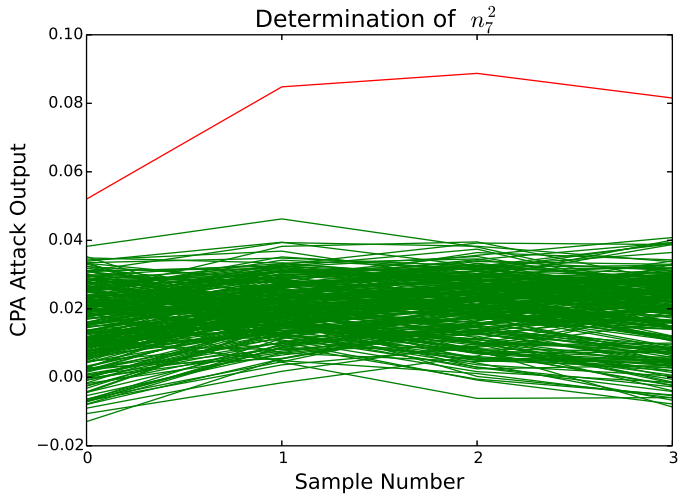

Figure 7.10: Determining n_7^2 means generating 256 candidate keys, which are tested by attacking a byte in the next round. This figure shows the correlation output of the CPA attack when attacking n_8^3 for the correct value of n_7^2 and n_8^3 in red. The incorrect values are displayed in green.

With n_7^2 and n_8^3 now known, we can continue with the CPA attack against $n_9^3, \cdots, n_{15}^3$. At this point we have an entire modified key:

$\mathbf{n^3}$ = [N N N N N N N N N N N N N N N N]

We can again apply the modified key $\mathbf{n^3}$ to the modified output of the previous round $\mathbf{m^2}$ to recover the complete output of round $r = 3$, which will be the actual input to round $r = 4$. This allows us to perform a CPA attack and recover the true round key $\mathbf{K^4}$. This round key can then be rolled backwards using the AES key schedule to determine the original encryption key.

We have now attacked an AES-CCM* implementation as specified in the IEEE 802.15.4 standard. This attack requires only the control of the four `FrameCounter` bytes, which are sent as plaintext over the air, as detailed in Section 7.2.1.

The computational load of the attack is minimal: performing these steps on an Intel i5-2540M laptop using a single thread program written in C++ takes under ten minutes with the 20 000 traces, using only the subset of points in each trace from Table 7.2. Note when performing the hypothetical value calculation for later rounds,

the calculation was accelerated using the Intel AES-NI instruction set for performing the $SubBytes()$, $ShiftRows()$, and $MixColumns()$ operations, which form part of a single AES round executed by this instruction [56].

7.4 Attacking Wireless Nodes

In the previous sections, I demonstrated the vulnerability of an IEEE 802.15.4 SoC device to power analysis, and how the AES-CCM* mode used during reception of an encrypted IEEE 802.15.4 packet can be attacked when the underlying hardware is vulnerable to power analysis. The last two aspects of this attack are to (1) demonstrate how we can trigger that encryption operation, and (2) determine where in the power signature the encryption occurred. This section demonstrates the ability of an attacker to perform these operations, and thus gives a complete attack against an IEEE 802.15.4 wireless node.

In Section 7.2.1 I detailed the message format that would cause an IEEE 802.15.4 wireless node to automatically decrypt the message on receipt. To validate this I used the same Dresden Elektronik RCB128RFA1 V6.3.1 board as in Section 7.1, programmed with Atmel's IEEE 802.15.4 stack version 2.8.0. I used the "Secure Star Network" example application for this, which initializes a standard IEEE 802.15.4 networking using security.

In order for the side-channel attack to be successful, the attacker needs to determine *when* the AES encryption is occurring. As a starting point, the attacker can use information on when the frame should have been received by the target node. Practically, this would be either the attacker's transmitter node toggling an IO line when the packet goes over the air, or the attacker could use another node that also receives the transmitted messages to toggle an IO line.

To determine the reliability of such a trigger, we measured the time between the frame being received and the actual start of AES encryption on the target node. Over 100 transmitted frames the delay varied between 311 and 338 μs. The mean value of the delay was 325 μs, with a standard deviation of 7 μs.

The jitter in the delay is assumed to be due to the software architecture, which uses an event queue to process the frames. Practically, the issue of aligning or resynchronizing power traces before applying power analysis is well known, and a number

of solutions have been proposed, such as comb filtering or windowing [33], differential frequency analysis [52], dynamic time warping [136], and principal component analysis [18].

To test the ability of an attacker to realign captured power traces, I used a simple normalized cross-correlation algorithm [77] to match a feature across multiple power traces for realignment, using the `scikit-image` implementation of this feature matching. This is an example of a *static alignment* method [80].

The selected feature was a window at 9.2–29.2 μs after the start of the AES encryption in one reference trace, meaning the matched feature extended slightly beyond the actual AES encryption. A plot of the output of the cross-correlation for different offsets of the template against another trace is shown in Fig. 7.11. I confirmed that a high correlation peak was generated only for a single sample around the AES algorithm with many sample power traces. A threshold of 0.965 on the correlation output (determined empirically) was used; if a power trace had no correlation peak higher than this level, the trace was dropped. This eliminates problems with a particularly noisy trace being matched incorrectly.

The successful realignment of traces is an important step when attacking real systems. As an example of trace realignment on another platform, see the attack on DESFire [109] which demonstrated how differential frequency analysis was used as a preprocessing step for CPA attack.

In a similar manner, it was demonstrated in [38] that it was possible to detect the location of software AES encryption based on the transmitter output of a wireless node coupled with further signal processing.

Future work on this IEEE 802.15.4 attack can include applying more advanced preprocessing techniques (such as differential frequency analysis or principal component analysis). But such preprocessing techniques are not required to fundamentally prove that (a) the AES core is leaking, and (b) the AES operation has some unique signature allowing realignment to succeed.

7.5 Summary

The IEEE 802.15.4 wireless standard is a popular lower layer for many protocols being used in or marketed for the coming "Internet of Things". Such protocols often use

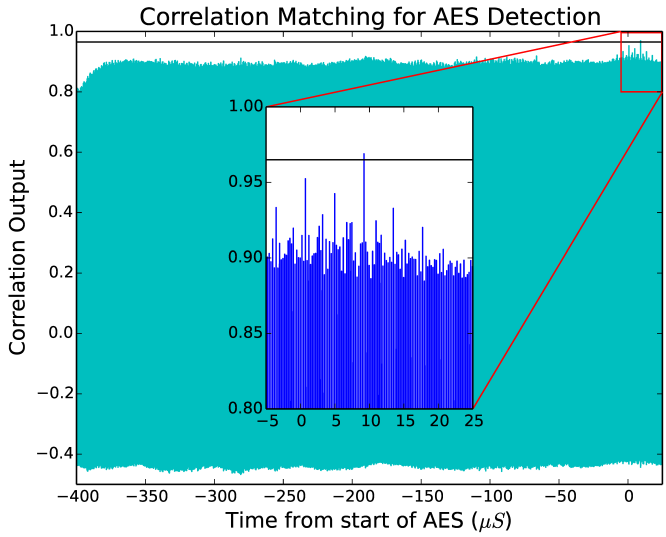

Figure 7.11: Correlation can be used to match a template from the power signature to align traces in the time domain, here the correct match should be at 9.2 μs, the location of the largest correlation peak.

the same underlying AES primitive as the IEEE 802.15.4 layer for security purposes.

This chapter has demonstrated significant vulnerabilities in a real IEEE 802.15.4 wireless node. A successful attack against the AES peripheral in the ATMega128RFA1 device was demonstrated. This attack was demonstrated against AES-ECB; as electronic code book (ECB) is not the operating mode of AES used in the network, we extended a previous attack on AES-CTR mode [64] to work against the AES-CCM* mode used in IEEE 802.15.4. This demonstrated that it is possible to recover the encryption key of a wireless node using side-channel power attacks and valid IEEE 802.15.4 messages sent to the node.

While this attack targeted a specific IEEE 802.15.4 SoC from Atmel, I believe similar attacks would be successful against AES peripherals on other devices. Users of IEEE 802.15.4 wireless networks, including users of protocols running on top of IEEE 802.15.4, need to seriously consider their security requirements in light of these attacks.

An additional type of attack against AES-CCM has been presented in [117], which builds on the work presented in this chapter. This demonstrates an attack against a commercially available ZigBee product (Philips Hue smart lights).

Chapter 8

Crowbar Glitch Generation

As mentioned in Chapter 1, fault or glitch attacks are a powerful tool for attacking embedded systems. This has been known since at least 1996, when A. Ross and M. Kuhn extensively demonstrated both clock and voltage glitches [5].

This chapter focuses on the practical aspect of injecting faults into a commercial off-the-shelf (COTS) embedded computer. Previous work has demonstrated the use of clock glitching or EM glitching on COTS embedded computers, such as attacking the Beaglebone Black using EM glitching as demonstrated in [60]. This work instead uses power supply glitching to insert faults in COTS embedded computers. Clock glitching will not work on more complex devices (discussed in section 8.1), and EM glitching is very sensitive to setup and equipment (discussed in section 8.1). A novel method of reliably introducing faults using power-supply glitching will be presented, a method that is applicable to a wide range of platforms and devices.

The novel method of generating power supply glitches uses a *crowbar* circuit, which aggressively shorts the power supply of the device to generate faults. This introduces ringing in the power distribution network on the circuit board, which propagates into the on-chip power distribution network. Ringing in this on-chip network is known to cause faults in digital devices, as shown in [149].

It will be demonstrated that a power supply glitch can be used to glitch a specific instruction. Previously it was considered that clock glitching could achieve much better temporal accuracy than power supply glitching[17], but it will be demonstrated that it is possible to achieve high temporal accuracy with power supply glitching on embedded systems.

This fault insertion is first characterized on a custom board using an AVR 8-bit microcontroller, then demonstrated on several COTS embedded computer boards: a Raspberry Pi running Linux, a Beaglebone Black running Linux, and an Android smart phone. It will also be demonstrated that it is possible to glitch an application

running on Linux or Android without causing a crash of the operating system or other negative effects. The applicability of this method to fault injection against Field Programmable Gate Array (FPGA) targets will also be demonstrated.

8.1 Related Work

The related work on cryptographic attacks may broadly be broken into two categories: methods of attacking algorithms using injected faults, and methods of injecting the faults on physical devices. Papers may often cover both categories: a new attack using fault injection is proposed, and this method is tested on a physical device. An excellent summary of papers in both these categories is given in [17] and in [107].

Three main methods of injecting faults are compared here: clock glitching, power glitching, and electromagnetic (EM) glitching. The reader is referred to [6, 14, 17] for other available methods. A summary of work relevant to this chapter for each of those three injection techniques will be presented next.

Clock Glitching

Clock glitching involves inserting additional rising edges into the input clock of the device, with the objective of violating timing constraints in the target device. For this to function, the clock must be used directly by the internal core. This means clock glitching will not be effective against two large classes of devices: those using internal oscillators, and those that use a Phase Lock Loop (PLL) to derive a new clock from the external clock. The majority of high-performance devices fall into the latter category, as they will run the internal core at a much higher frequency than the external clock.

Clock glitching on an Atmel AVR microcontroller is throughly presented in [10], which uses the same microcontroller family as being used in this chapter. In addition an extensive case study of clock glitching has been presented in [75] that used perturbations in the device power supply to improve the effect of the clock glitches, but did not consider the effect of power supply glitching alone. In [75] two devices are targeted: an ARM Cortex-M0 implemented in a NXP LPC1114 device, and an Atmel ATxmega 256 device.

These two papers demonstrate that with fine-grained control of the glitch timing, various instruction and data movements can be faulted with fine-grained control over the fault result.

EM Glitching

A typical EM glitch injection setup involves a precision X-Y table that can position the probe over the surface of the target chip. It has been demonstrated that for a successful glitch injection a very high precision is required when placing the probe over the chip surface [60, 95]. In addition if Package-on-Package technology is used in the target chip, this can make glitching more difficult, as a memory die has been stacked over the processor die [60].

EM glitching can achieve very fine-grained control over the fault effect, for example attempting to fault operations of specific registers [60]. EM glitching is a very powerful attack, but has the downside of requiring a more complex physical environment.

Power Glitching

Power glitching involves manipulation of the power supply of the target devices to generate faults; a simple example is how lowering the supply voltage will again introduce timing errors due to increased propagation delay. This *underpowering* has proven to introduce faults in ARM-9 devices during cryptographic operations [15]. This does not however provide good temporal accuracy, making it difficult for the glitch to target specific instruction.

The ability to target specific instructions can be achieved by instead inserting a 'spike' in the power rail at a specific instance in time. Both positive and negative voltage spikes can be inserted into the external power rails, where the spikes have a narrow width and attempt to cause faults in specific instructions. Both positive and negative spikes on the external rails result in similar waveforms internally in the target device, as demonstrated in [149].

A comparison of voltage glitching on three targets is given in [27], including attacking a 'secure' device. The authors of [27] provide a search methodology for determining ideal parameters of a glitch, i.e. finding the glitch amplitude and width. The results presented in [27] are extremely useful in visualizing the sensitivity of a

system to a voltage glitch.

A comprehensive discussion of voltage glitching against FPGA target has been presented in [24], where the authors compared voltage glitching to laser (optical) glitching. In that work voltage glitching is shown to be effective against FPGAs for fault injection, and voltages in the range of 45V – 80V were found to be most effective for their experimental setup.

8.2 Glitching Mechanism

The glitch mechanism explored in this chapter is a simple 'crowbar' circuit. This circuit applies a short across the power rails of the target device, the specific waveform generated depending on the target device power supplies.

The glitch is generated with an N-Channel MOSFET (IRF IRF7807), driven using the glitch generation circuitry from the ChipWhisperer hardware described in Chapter 4. The selected MOSFET is a higher-power logic-level MOSFET with $88A$ of peak pulse current capability and 0.014Ω $R_{DS(ON)}$. As is typical for such a MOSFET, the gate charge is sufficient that generating very narrow glitches requires more care in the design of the driver circuit [12].

If very narrow glitches are required, a lower-power MOSFET (such as IRF IRLML2502) can be used, as this device has lower gate charge requirements, and can be switched faster than the higher-power MOSFET. This particular MOSFET has a $R_{DS(ON)}$ of 0.035Ω, meaning it would be less effective against low-impedance power rails likely to be found on high-speed processor boards.

8.3 Target Devices

The glitching attack is demonstrated against five targets: four microcontroller/microprocessor devices, and one FPGA device. The first target is a simple 8-bit microcontroller, the next three are various types of ARM-based System-on-a-Chip (SoC) devices, and the final target is a Xilinx Spartan 6 FPGA. The SoC devices are selected to represent those found in a wide variety of embedded systems, from single-board Linux computers to standard smartphones.

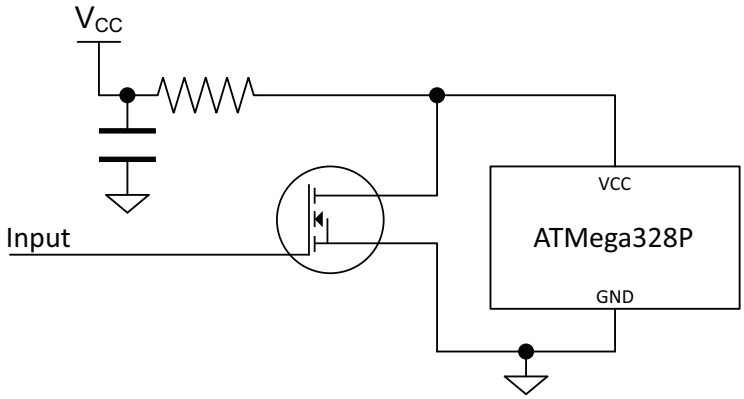

Figure 8.1: The crowbar circuit using an N-Channel MOSFET is connected across the AVR power pins, and also allows power measurement across a shunt resistor.

8.3.1 AVR Microcontroller.

As the AVR microcontroller has been studied for clock glitching ([10]), and power glitching ([120, 27]), it serves as a useful benchmark for this work. I specifically use the ATMega328P AVR microcontroller in DIP package running from a 7.3728 MHz crystal oscillator.

The glitching attack against the AVR uses the lower-power MOSFET (part number IRLML2502), connected as shown in Fig. 8.1. The series resistor serves two purposes: first, it allows the lower-power MOSFET to clamp the supply voltage towards zero, and second it allows simultaneous power-analysis, including triggering the fault based on patterns in the power consumption waveform.

8.3.2 Raspberry Pi (ARM11)

The *Raspberry Pi* is a low-cost single-board computer with an ARM11 based single-core processor, the BCM2835 from Broadcom, and runs at 700 MHz core frequency. This platform was loaded with Linux Debian with kernel 3.12.28.

For the power-glitching attack, I used the higher-power MOSFET IRF7807 connected across 220 nF decoupling capacitor C65, that capacitor being part of the VDD_{CORE} power distribution network. Additional details of the hardware setup are available as part of a tutorial[1].

[1]Details are posted as part of the ChipWhisperer Documentation, available at `http://www.chipwhisperer.com`

8.3.3 Beaglebone Black (ARM Cortex-A8)

The *Beaglebone Black* is a low-cost development board with an ARM Cortex-A8 based single-core processor, the AM3358 from Texas Instruments (TI), and runs at 1 GHz core frequency. This is the most powerful platform tested in this chapter, and runs Linux Debian with kernel 3.8.13-bone47.

This platform was selected in particular as it is also used as an EM glitching target by Hummel in [60]. Hummel extensively characterized the results of EM glitching over the surface of the main processor, and noted that careful positioning of the glitching coil was required to avoid simply rebooting the target.

The power-glitching attack again used the higher-power MOSFET connected across 100 nF decoupling capacitor C63, part of the VDD_{MPU} rail. Note that attacks when the crowbar was connected across the VDD_{CORE} network were unsuccessful, only attacks against the VDD_{MPU} succeeded.

8.3.4 Android Smart Phone (ARM11)

A *HTC Wildfire S* smart phone was used with the stock image for this phone (Android 2.3.3). The main System-on-a-Chip (SoC) in this phone is a Qualcomm MSM7227. This is a highly integrated device with an ARM11 applications processor, applications DSP, ARM9 baseband processor, and baseband DSP. The user code will run in the ARM11 applications processor, which has a clock speed of 400 MHz.

The crowbar is attached across the capacitor shown in Fig. 8.2. This capacitor appears to be part of the power distribution network for the application processor core, based on comparison of the voltage at this point to the known core voltage of the device.

8.3.5 FPGA Board (SAKURA-G)

For this chapter the SAKURA-G board[58] is used as a platform for fault injection. This board contains a Spartan 6 LX75 FPGA (part number XC6SLX75-CSG484 in 2C speed grade) along with supporting circuitry. This board is designed for side-channel analysis so does not have capacitors mounted on the VCC_{INT} power rail, and contains a shunt resistor across this power rail. The lack of decoupling capacitors

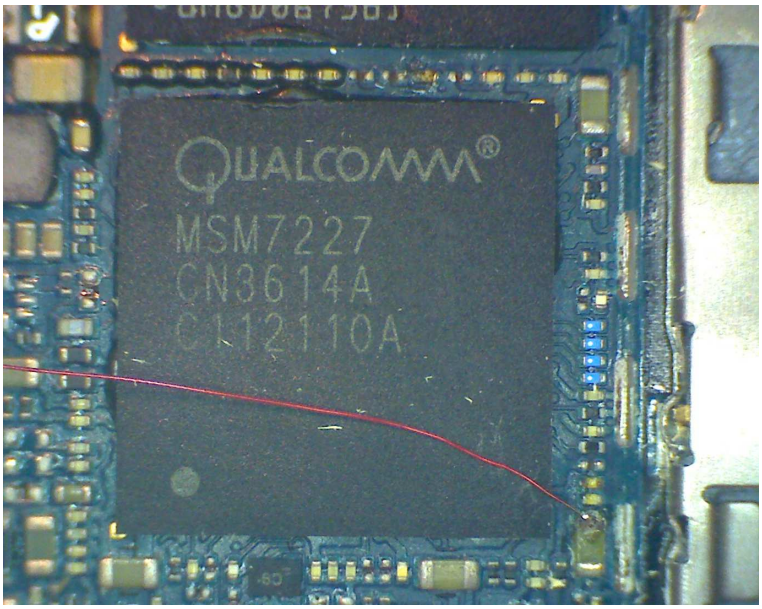

Figure 8.2: A small length of magnet wire is used to connect a capacitor around the MSM7227 device to the crowbar MOSFET for glitch insertion. The ground connection comes from another point closer to where the MOSFET is mounted.

on this rail suggests the fault waveform should have little ringing when the crowbar is released.

The higher-power MOSFET (IRF7807) is used to short the VCC_{INT} power rail for the FPGA. As the SAKURA-G board contains a SMA connector on the VCC_{INT} rail, the MOSFET can be connected across this connector (J2).

8.4 Fault Insertion Results

Two types of faults were tested: in the first a simple code sample that should be highly sensitive to faults was tested, and in the second we explored faulting specific operations or data within algorithms. We refer to the first as a 'low-precision fault', as timing of the fault does not have a precise temporal trigger – the fault is being inserted at a random point during the clock cycle of the device. Low-precision fault insertion uses a fixed pulse width to activate the crowbar circuit.

For faulting specific operations or data, we use a 'high-precision fault', where specific temporal relationships between activity on the target device and fault injection time are maintained.

Listing 8.1: This code should result in `25000000-5000-5000` being printed for every successful loop.

```
int i,j,cnt;
while(1){
for(i=0; i<5000; i++){
for(j=0; j<5000; j++){
cnt++;
}
}
printf("%d-%d-%d\n", cnt, i, j);
}
```

8.4.1 Low-Precision Faults on Microprocessors

For this low-precision work, the code being glitched is given in Listing 8.1. This was based on previously published glitching examples in [27]. The objective of the glitch is causing an incorrect count for the variable `cnt`. I do not explore the specific cause of the glitch (i.e. what instruction or data is being affected), only the resulting output was incorrectly calculated (i.e. a fault was inserted at some point).

This code is used on the four microcontroller / microprocessor targets. Discussion of low-precision faults on FPGA targets will be given in Section 8.4.2.

On the AVR target, Listing 8.1 is compiled directly onto 'bare metal' – there is no OS, only Listing 8.1 is running, with the `printf()` statements sending data over the serial port.

Listing 8.1 was compiled as a regular user program on both the two Linux-based system and the Android system. The underlying OS will still be running background processes, and our objective is only to fault the user program. For the two Linux systems we interact with the user program via a remote `ssh` terminal over the Ethernet connection, and with the Android system we interact using the touch-screen interface. The Android system uses a Java version of Listing 8.1.

This fault has been successfully applied against all four of the processor target devices described in Section 8.3. A successful fault is one where a single outer loop of the program from Listing 8.1 produces an incorrect result. The program must continue

Table 8.1: Low-precision fault injection is used against all of these devices to cause the code from Listing 1 to calculate an incorrect result.

Target	Crowbar Activation Time
ATMega328P	135 nS
Raspberry Pi	635 nS
Beaglebone Black	485 nS
Android Smartphone	615 nS

to run after the incorrect calculation without crashing. Details of the parameters for a successful fault are given in Table 8.1 for each target device.

As the fault is inserted at a random point in time, the only parameter to vary is the pulse width. The test program on the AVR has exclusive use of the core, as there is no OS, so a randomly inserted fault is almost certain to occur around a sensitive operation. On the Linux and Android system the underlying OS and other processes are also running, but due to the use of a infinite loop the test program will monopolize a single core, making it very likely a randomly selected point in time will result in a fault inserted into our sensitive code rather than crashing the OS or a background process.

An example of the fault waveform for the Raspberry Pi device is given in Fig. 8.3. It can be seen the fault waveform involves both the power drooping while the crowbar is activated, along with substantial ringing once the crowbar is released. On this specific target it seems likely the fault method may be the voltage regulator responding to the sudden current change as the crowbar is released.

The output of the software from Listing 8.1 running on the Raspberry Pi during a fault injection is shown in Fig. 8.4, and the Android Smartphone shown in Fig. 8.5.

This work does not characterize which aspects of this waveform are critical to fault generation, but instead simply parameterizes the fault based on length of time the crowbar is activated. The level and frequency of the ringing generated when the crowbar is released depends greatly on the power distribution network (PDN) design (including for example circuit board layout and number of decoupling capacitors), along with the location where the crowbar is connected across.

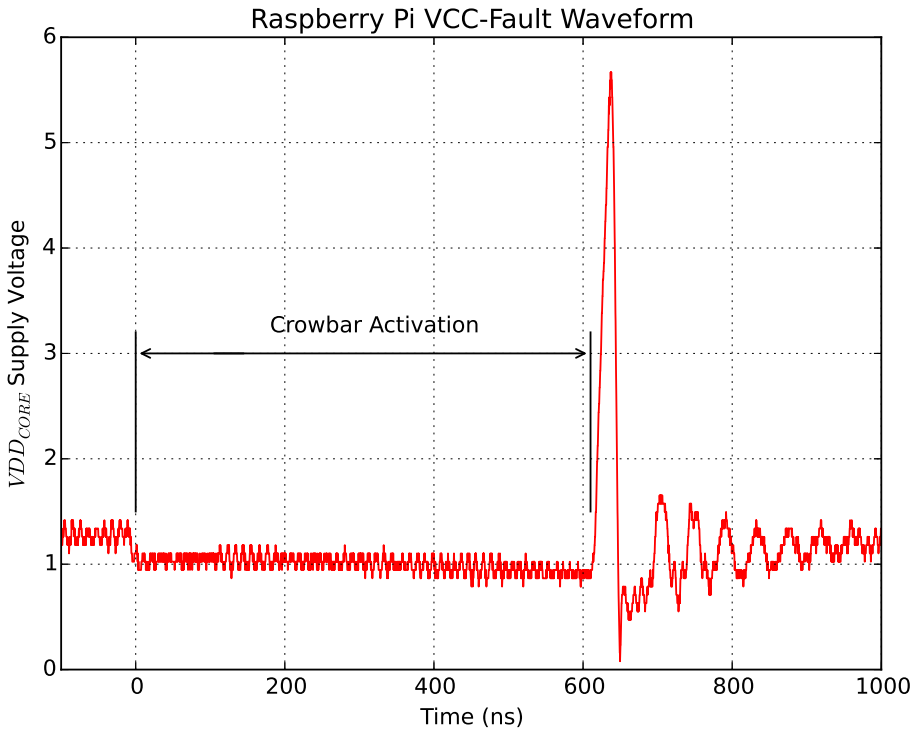

Figure 8.3: The signal on the VCC_{CORE} rail for the Raspberry Pi during fault injection.

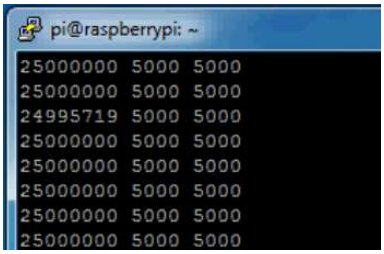

Figure 8.4: An implementation of Listing 8.1 in C was used in a Linux application for testing purposes on the Raspberry Pi and Beaglebone Black. The output is monitored via a `ssh` connection, which is done to ensure the OS and network connection does not crash during the fault injection.

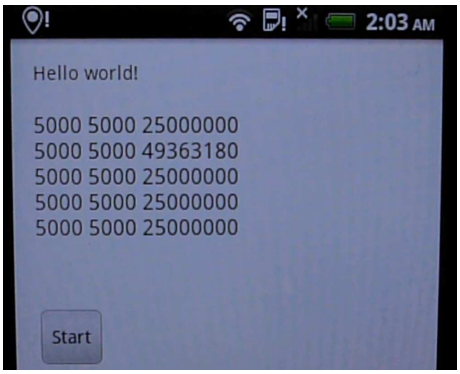

Figure 8.5: An implementation of Listing 8.1 in Java was used in a simple Android application for testing purposes. The injected fault causes an incorrect count for a single loop iteration (49363180 instead of expected 25000000).

8.4.2 Low-Precision Faults on FPGAs

Fault injection on FPGAs has many uses, from simulating errors such as are expected from high-radiation environment[68] to attacking cryptographic implementations built on FPGA systems[70]. Work on the former has shown for example how to determine what specific type of errors occurred as a result of radiation-induced faults in an FPGA [141], and methods of simulating[28] or emulating[3] single-event upsets.

As mentioned, this work uses the SAKURA-G board [58] with a crowbar against the VCC_{INT} rail. As expected due to the lack of decoupling capacitors, the crowbar insertion has a very 'clean' waveform, as can be seen in Fig. 8.6. There is almost no ringing as a result of releasing the crowbar.

FPGA Design

A basic design consisting of sixteen separate 32-bit registers is instantiated in the FPGA. Eight of these registers are loaded with all `1's` based on an external reset signal, and the other eight of these registers are loaded with all `0's` when that external signal is asserted.

The status of the registers are monitored by two external pins – this is able to detect one or more bits flipping from 0 to 1 (bit-set fault), or from 1 to 0 (bit-reset fault). An additional input signal temporarily overwrites the register value, used as a self-test to confirm the fault detection logic is still functioning.

As the configuration data of the FPGA itself is stored in SRAM and subject

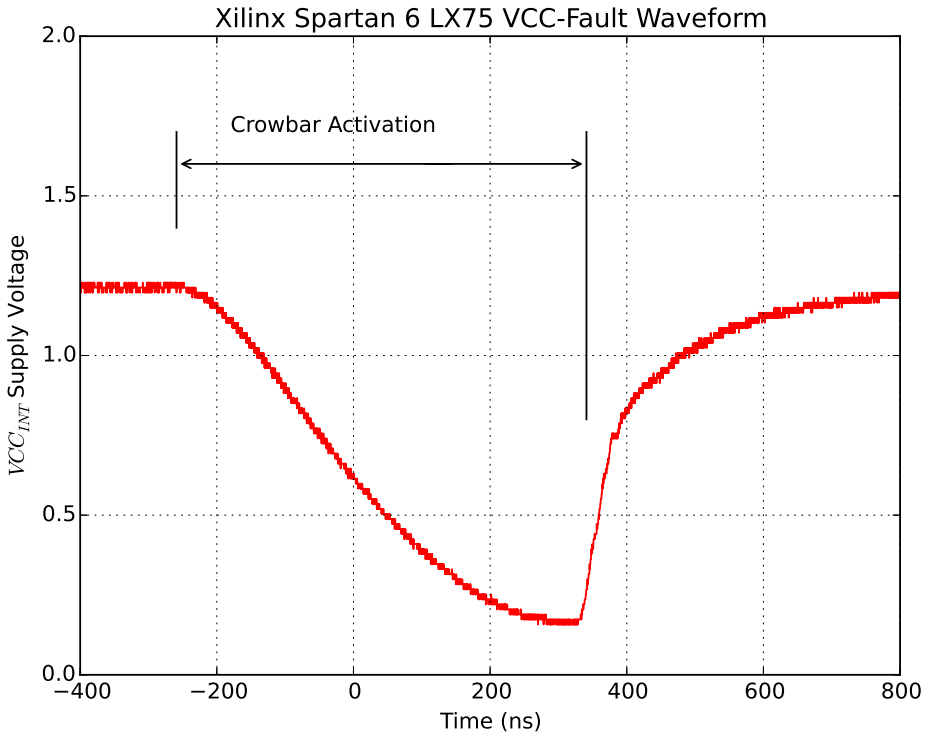

Figure 8.6: The lack of decoupling capacitors on the SAKURA-G board along with inclusion of resistive shunt

to corruption, the configuration data itself may become corrupted when inserting a fault[142]. A fault that is able to be cleared by asserting the external reset signal is considered a temporary fault (labeled a 'Design Register Fault' in the results from Table 8.2). If the design fails to function after the fault insertion even with an external reset, this is considered a 'Functional Failure'.

Determining that a 'functional failure' has occurred only means the configuration data specific to this design has been corrupted in such a way to prevent the design from working. It is also necessary to determine if other bits of the configuration data has been corrupted to properly characterize the fault injection results. These other configuration bits are portions of the FPGA that are not being used in the current design.

To accomplish this, the continuous CRC-check feature of the Spartan 6 FPGA can be used. This feature causes the FPGA to set the `INIT_B` pin to a logic low when the configuration memory of the FPGA changes. Monitoring this pin determines when a 'CRC Failure' has occurred, indicating the configuration data of the FPGA has been changed by the fault [146].

Once a 'CRC Failure' is detected, the readback feature of the FPGA is used to determine how many bits have flipped. A reference bitstream is first created based on a correctly loaded FPGA, and this reference is then compared to the new read-back file from the FPGA with a CRC failure. Based on the difference between these files the specific number of bits corrupted in the FPGA bitstream can be determined.

Fault Results

The results of various crowbar activation times on faults in the FPGA is given in Table 8.2. If the crowbar is activated longer than 900 nS, the FPGA enters a reset state and attempts to reload the configuration data.

For small fault injection widths ($\leq$550 nS), the configuration data of the FPGA is only occasionally corrupted (1 of 10 fault attempts causes at least one bit of configuration data corruption at 550 nS). Crowbar activation widths of 600 nS or greater always result in at least one bit of corruption of the configuration data stored in the FPGA. The number of bits corrupted tends to increase non-linearly with relation to crowbar activation time. The total FPGA readback bitstream has 2 452 898 bits, so

Table 8.2: Results of fault injection against Spartan 6 LX75 FPGA, repeated 10× for each width.

Width	SRAM Configuration Data Faults			Design Register Faults		
	CRC Failures	Functional Failures	Avg Bit Diff. of Failure	Set	Reset	Set & Reset
550 nS	1	0	1028	0	0	0
600 nS	10	0	1037	0	0	0
650 nS	10	1	1050	0	0	0
700 nS	10	0	1052	0	0	0
750 nS	10	0	1695	0	2	3
800 nS	10	0	7269	0	1	2
850 nS	10	0	20201	0	1	2
900 nS	10	8	40026	0	2	0

for example if 1028 bits are corrupted this represents 0.042% of bits corrupted. A graph showing the relationship between glitch length and number of bits corrupted is provided in Fig. 8.7.

Assuming the objective is to insert a fault into the registers inside the FPGA design, it can be seen there is an optimal glitch width that minimizes the amount of corruption within the configuration data, while still causing the values of registers to change inside the FPGA design. In this specific design a glitch width of about 750 nS would frequently (50 % of the time) result in one or more bit flip(s) in the register(s) without noticeably damaging the FPGA design. Note there *is* some corruption of the FPGA design, but the 'damage' is sparse enough to make a functional failure in the design unlikely.

These results demonstrate it is possible to use a crowbar fault mechanism on a FPGA to introduce random faults into both the configuration information and the registers used in the working FPGA design. Previous work on voltage fault attacks against FPGAs reported the ability to cause bit-flips in registers used within the FPGA design, but not modify the configuration information[24].

8.4.3 High-Precision Faults

The high-precision fault insertion uses a more complex fault waveform, shown in Fig. 8.8. This fault waveform is capable of activating the crowbar circuit for fractions of the clock cycle, and with precise timings from edges of the device clock. This

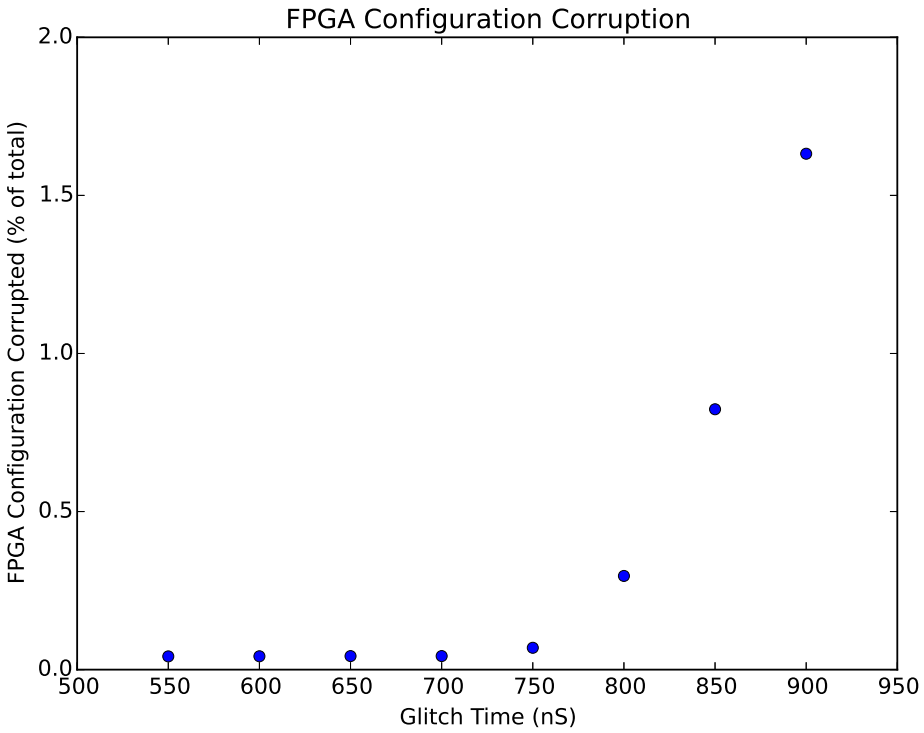

Figure 8.7: Amount of FPGA configuration data corrupted based on length of crowbar activation.

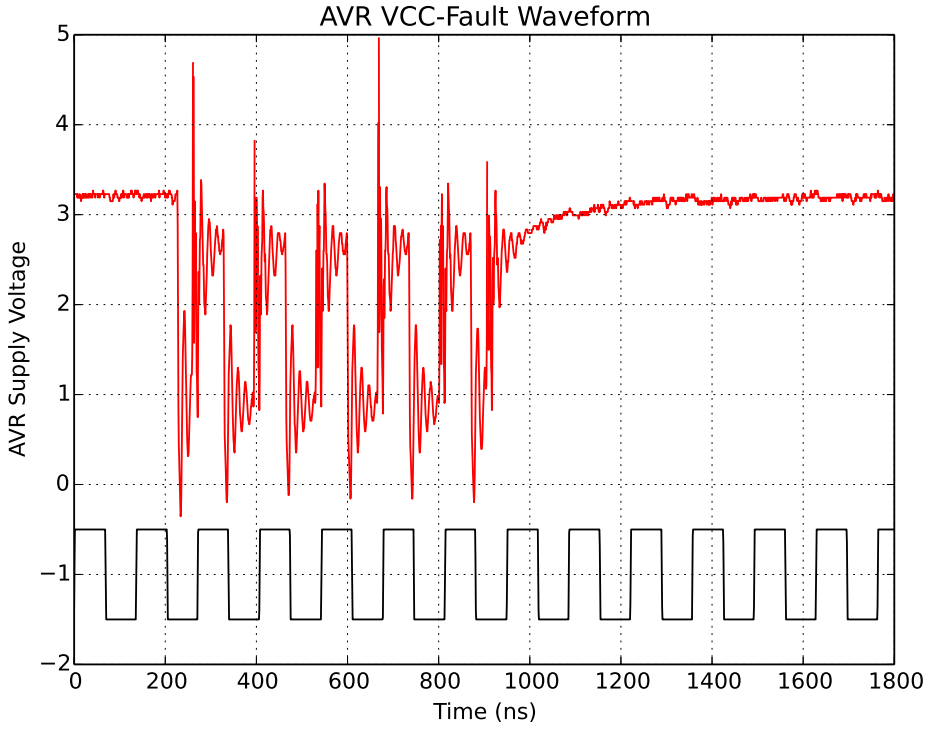

Figure 8.8: The signal on the VCC pin when performing high-precision fault injections on the AVR is given in red. The black waveform is the 'glitch clock', which is four times the device clock.

requires access to the device clock to maintain synchronization, but it does not require the ability to manipulate the clock.

While this work did not explore high-precision fault attacks on a device with an internal oscillator or PLL, previous work has demonstrated the ability of a simple circuit to perform the clock recovery when no external oscillator is available [103]. Thus the work in this section should also be applicable to devices with internal oscillators or PLLs, where clock-glitching attacks are not possible.

The high-precision fault injection uses a trigger signal from the target device. The trigger signal indicates when the target device is performing the sensitive operating we wish to fault. For timing the crowbar activation, a 'fault clock' is generated that is phase-locked to the device clock, but operating at four times the frequency of the device clock.

This means the following parameters can be adjusted for each fault operation:

Starting Offset: After the trigger occurs, the number of cycles of the glitch clock before the crowbar is activated. This can be seen as starting the glitch during one of four phases of the device clock (as the glitch clock is four times the device clock).

Cycles Glitched: Number of cycles of the glitch clock during which the crowbar is activated. Note from Fig. 8.8 the crowbar is only activated for a portion of each cycle.

Phase Offset: The delay from the rising edge of the glitch clock to the crowbar being activated for that cycle. A positive offset indicates it is activated after the rising edge, a negative offset indicates before the rising edge.

Glitch Length: The length of time the crowbar is activated for within each cycle.

Three different code samples are used for fault injection. These samples are designed to test bit-set and bit-reset faults, along with exploring modifying single or multiple bytes within an operation (such as when targeting a specific byte of the AES state).

The code used for detecting bit-set faults is given in Listing 8.2, and bit-reset faults is given in Listing 8.3. Both of these samples are designed to use both SRAM and registers, along with repeating the operation over multiple clock cycles.

The results of varying the starting offset, cycles glitched, and phase offset is given in Fig. 8.9 and Fig. 8.10 for bit-set and bit-reset faults respectively (these figures appear at the end of this chapter). The glitch length was fixed at 16.9 nS (50% of the glitch clock period). For each combination of parameters the output of the code sample given in either Listing 8.2 or Listing 8.3 is compared to the expected output. In addition to detecting either single-bit or multi-bit faults, reset of the device (via printing of a start-up sequence) is detected.

The phase offset is varied from $-108°$ to $108°$ in $1.8°$ steps. The cycles glitched is varied from 4 to 40 cycles in 2-cycle steps. For all test cases the device is powered off and on after each fault attempt. This is to avoid errors caused by the device entering a lockup or failed state, or if some unknown faults have been introduced that would affect future tests. Powering the device completely off and on achieves a reliable known-state for each test to be performed on.

These figures demonstrate that selecting the phase offset is a critical parameter

Listing 8.2: Passing `0x00` and `0x00` for both `a` and `b` allows this code to detect bit-set faults. As `a` is declared volatile the value is loaded from and saved to SRAM after each OR operation as shown in the resulting assembly code.

```
uint8_t glitch_bitset(volatile uint8_t a, uint8_t b) {
trigger_high();
a = a | b;
//Each OR operation compiles to the following ASM:
//ldd  r24, Y+1
//or   r24, r22
//std  Y+1, r24
a = a | b;
...
a = a | b;
a = a | b;
return a;
}

result = glitch_bitset(0x00, 0x00);
```

Listing 8.3: Passing `0xFF` and `0xFF` for both `a` and `b` allows this code to detect bit-reset faults.

```
uint8_t glitch_bitreset(volatile uint8_t a, uint8_t b){
trigger_high();
a = a & b;
a = a & b;
...
a = a & b;
a = a & b;
return a;
}

result = glitch_bitreset(0xFF, 0xFF);
```

for a successful fault insertion. Tuning of this parameter allows insertion of either single-bit or multi-bit faults in both the bit-set and bit-reset fault case.

To extend this to multi-byte operations, the code from Listing 8.4 is used. This code applies similar functions to those used in many cryptographic operations, but does not differentiate from bit-set and bit-reset faults.

This attack fixes the phase offset and cycles glitched parameters based on those discovered from the single-bit fault operations, in this case around 72 degrees phase offset and 5 cycles glitched.

The starting offset is then varied to attempt targeting of specific bits and bytes within an 8-byte array. The fault attempt is repeated for each starting offset 20 times, in order to determine the reliability of the fault operation.

As can be see in Fig. 8.11, faults can be targeted against specific bytes within the array operation. Specific starting offsets have close to 100% reliability on fault insertion (this figure appears at the end of this chapter).

8.5 Discussion

This section discusses the applicability of crowbar fault injection to real-world platforms.

8.5.1 Generating Fault Signals

The crowbar attack method requires a very simple fault signal. To replicate the results from Section 8.4.1, one only requires a pulse generator to drive the MOSFET. This signal can even be generated by a simple microcontroller if a laboratory pulse generator is not available. This makes it possible to add a fault generator to a system (such as connecting to a trusted computing module inside a laptop), while the user of the system is unaware of the fault generators presence. The attacker may choose to activate the fault module at a later point in time, or only have the module active during specific sensitive operations.

Replicating the results in Section 8.4.3 is easiest when using a FPGA-based system. This work uses the ChipWhisperer platform from Chapter 4, but almost any FPGA board is capable of performing the require clock multiplication and shifting used to generate the fault signal. As was mentioned in the results section, the fault waveform

Listing 8.4: By comparing the value of array `a` after the call to `glitch_mb()` we can detect the location of faults across several bytes. Note certain bits are only sensitive to bit-set and certain bits are only sensitive to bit-reset faults.

```
void glitch_mb(uint8_t * a, uint8_t * b){
  trigger_high();
  for (uint8_t i = 0; i < 8;  i++){
    a[i] ^= b[i];
  }
}

void run_test(void){
  uint8_t a[8],b[8];

  for(uint8_t i = 0; i < 8; i++){
    a[i] = 0xAA;
    b[i] = 0xFF;
  }
  glitch_mb(a,b);

  //Value of 'a' is now checked
}
```

was extremely sensitive to the location of the first faulted cycle, along with phase of the fault relative to the device clock edge.

8.5.2 Finding Vulnerable Supplies

When attacking a device, it is required to determine the vulnerable supply rail. Even very simple devices will typically have at least two power rails (analog and digital), but more complex devices such as SoC could have many more (such as processor, memory, USB, clock domain, and analog).

This work attacked three such SoC devices, with varying levels of public documentation. The Beaglebone Black had full schematics and documentation published, including details of the SoC device. The Raspberry Pi has schematics but no details of the SoC, and the Android phone had no schematics and no details of the SoC. Determining the sensitive rail for each device will be discussed in sequence.

On the Beaglebone Black, the schematic shows there are two power rails of interest: VDD_{CORE} and VDD_{MPU}. Attempting to use a crowbar on the VDD_{CORE} was not successful, where the crowbar was inserted on a number of different locations underneath the BGA package. By comparison using a crowbar against the VDD_{MPU} rail was successful on the first attempt. As VDD_{MPU} is the *MicroProcessor Unit* rail, we would expect this rail to be the sensitive rail.

The Raspberry Pi also had schematics available, but in this case the SoC only had a VDD_{CORE} rail. Glitching against a randomly selected decoupling capacitor from this rail was successful.

The Android phone presented the most difficulty in determining the sensitive supply. There is no public documentation for the main SoC (Qualcomm MSM7227) device, and of course no schematics for the phone. Probing the decoupling capacitors mounted around the device showed 2.6V, 1.8V, 1.25V, and 1.33V being present. Based on the layout the 1.8V capacitors were likely part of the memory interface, leaving the 1.25V and 1.33V rails. Ultimately I found the 1.33V rail, using the point from Fig. 8.2, was a vulnerable location for fault insertion.

8.5.3 Triggering Faults

It has been shown in Section 8.4.3 that a careful timing allowed crowbar fault injection to achieve extremely high reliability. For simplicity these tests repeated the instruction multiple times, but it can be noted that changing the timing by only a small percentage of the clock cycle resulted in different fault effects. In practice, this careful timing can be achieved by using either a trigger signal from the target device (in the case of instrumentation purposely added), or with a trigger based on a power consumption or I/O activity trigger of the target device[105].

8.6 Summary

We have introduced a novel method of injecting voltage faults into hardware devices using a MOSFET to short the power supply of the device with very precise control over timing of the faults. This is called the *crowbar* injection technique. The use of this technique against several platforms, including devices used in previous publications, has been presented. In addition several platforms are standard 'off-the-shelf' boards, showing how the crowbar technique can be used on real embedded systems.

The crowbar technique takes advantage of the properties of the power distribution networks on printed circuit boards to generate ringing in these networks. This ringing is presumed to perturb the power distribution networks on the target chip itself, which is known to cause faulty operations [149].

The use of fine control over the fault timing has also demonstrated that faults with very high reliability can be inserted, determining for example if a single- or multi-bit fault should be introduced, or to fault a single byte out of a larger array operation.

Currently this 'high-precision' faulting has only been demonstrated on simple 8-bit Atmel AVR microcontrollers. Future work is needed to test larger platforms such as embedded Linux computers to determine the reliability of high-precision fault attacks, and their ability to target very specific instructions or data.

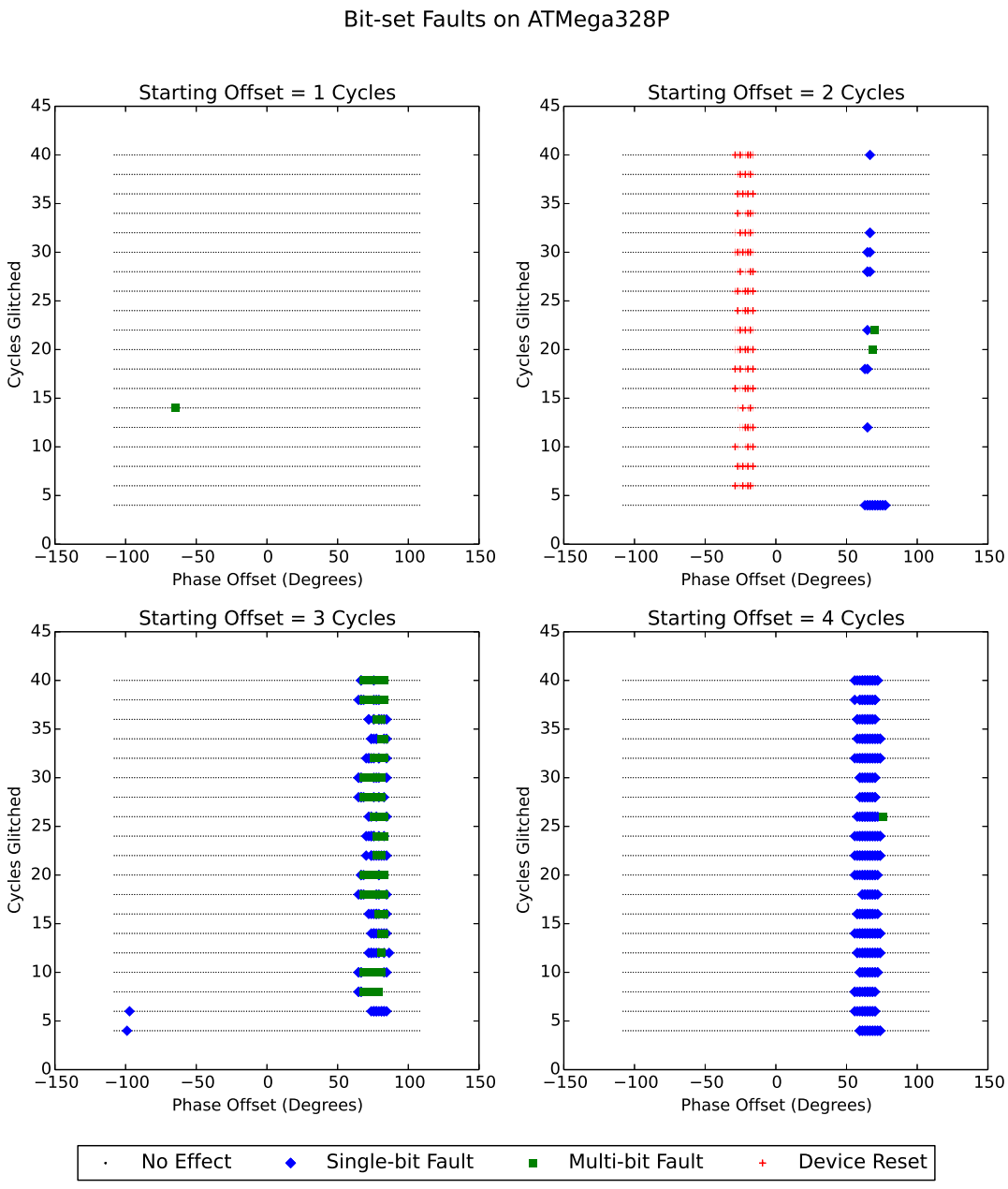

Figure 8.9: Bit-set faults mean at least one bit that should have been a '0' was read as a '1'. It can be seen both single-bit and multi-bit faults can be injected depending on the phase and starting offset.

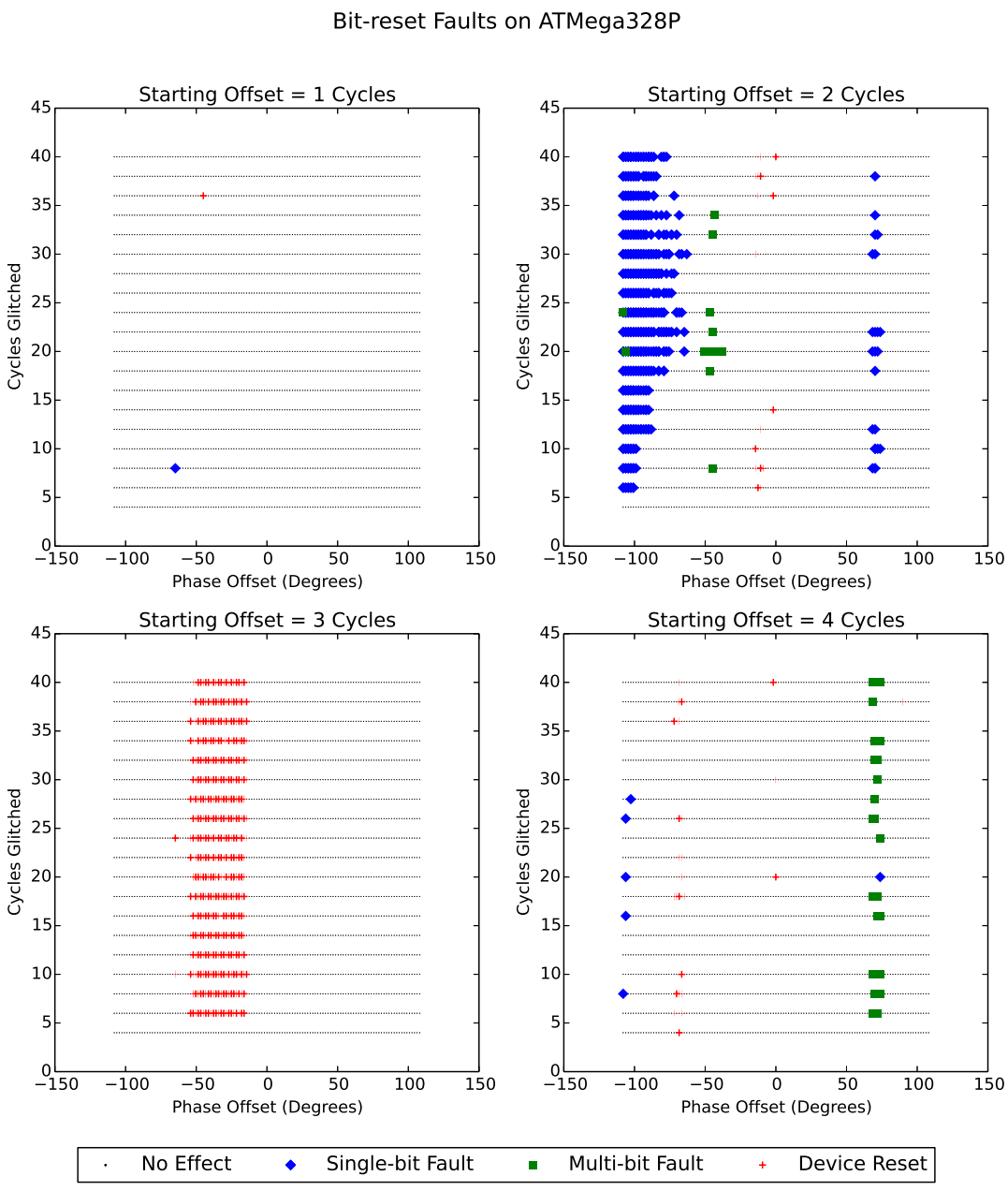

Figure 8.10: Bit-reset faults mean at least one bit that should have been a '1' was read as a '0'. It can be seen both single-bit and multi-bit faults can be injected depending on the phase and starting offset.

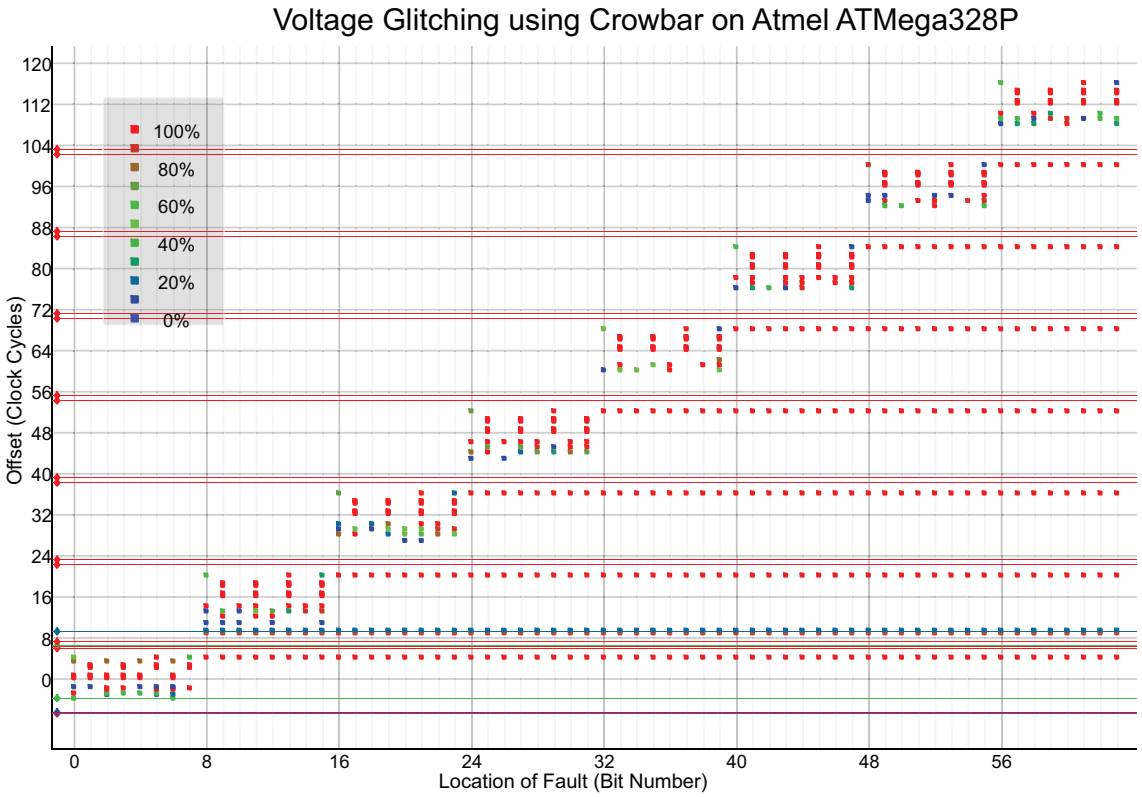

Figure 8.11: A 64-bit number is manipulated, and the reliability of fault insertion on each bit for different glitch locations is graphed. Squares indicate a single bit fault, horizontal lines indicate a device reset. The color of the square (or line) indicates empirical probability of that fault result for a given offset. For example at an offset of 16.25 clock cycles results in the same four bits located within the second byte of the copy operating being marked as incorrect for 100% of observations. As the data being copied is `10101010` binary, this may indicate only those bits set to '1' were affected by a bit-reset fault.

Chapter 9

Conclusions and Future Work

9.1 Concluding Remarks

Embedded computers already exist in many devices we own or use, and this trend looks only to accelerate in future years. These computers are becoming ever more powerful and interconnected, making them valuable targets for hackers. In the past few years ransomware has become a popular method of extorting consumers, by demanding payment to unlock their data [51]. Similar attacks may come to hardware devices – attackers have already demonstrated an ability to remotely reprogram embedded computers in cars, and it's not a stretch to imagine attackers using this to prevent the car from working unless a special unique key is entered. This key is produced to the consumer only after payment on a ransomware website.

Solving this problem requires fundamental shifts in the design process of embedded systems. Security must be a continuous consideration, and must be something that becomes part of the training of any engineer designing embedded systems. Accomplishing this requires a more fluid transfer of knowledge from academia – who has known about these attacks for 18 years – to industry.

The novel architecture for a new capture hardware, and the open-source ChipWhisperer project presented in this thesis, are another step towards this goal. The ChipWhisperer project is already used in a number of universities, by companies running internal training courses, and by open training courses that can be taken by embedded engineers.

This thesis also includes several demonstrations of the tool against hardware and software used in (or similar to those used in) real applications. Many devices include an encrypted bootloader for example, and I've worked to demonstrate this may be considerably less secure than marketing material suggests. Similarly, be breaking the hardware AES used on the ATMega128RFA1 device, it can help designers understand the true vulnerability of these devices. Only with this knowledge can the correct

decisions about system-level functionality such as how to distribute the AES keys and how often to update them be made.

The open-source ChipWhisperer fits within a variety of other open-source tools. In particular, other work such as the FOBOS [138] and GIAnT [108] have also worked on providing a platform for other researchers to use. The ChipWhisperer presented here more closely aims to bridge between academia and industry by providing an "all in one" solution including both hardware and software, such that working engineers can easily learn how power analysis attacks work and where they apply.

A considerably body of previous work has detailed experimental setup as part of work on power analysis, for example detailing oscilloscopes, sampling rates, and probes being used for a given paper. My work specifically looks at how sample rates and clock synchronization may change the results of attacks on specific targets. This can be useful when attempting to duplicate results published elsewhere, by providing information on the effects of various sample rates and capture strategies on attack results.

9.2 Future Work

The open-source ChipWhisperer project has only basic attacks implemented currently. Future work in implementing more advanced attack modules will continue to make this tool useful for both groups in academia and industry. In particular, demonstrating more advanced attacks may encourage authors of new work to directly contribute a working version of their attack. This would greatly help with the transfer of knowledge of academia to industry, as recreating an algorithm from published papers can be very intensive work.

Demonstrating the platform against more advanced devices is also of importance – for example demonstrating it against a multicore ARM device. This is of particular importance as the cost of such devices falls, and these devices will be more common in embedded systems. Demonstrations against higher-performance embedded systems are also required, such as very fast hardware AES cores.

Documentation on both the tool usage and the attacks themselves is also critical to help disseminate the knowledge of embedded hardware security threats and solutions. As part of my open-source ChipWhisperer project I've documented much of this in

hands-on tutorials at `http://www.ChipWhisperer.com`, but this is ongoing work to expand this knowledge base.

9.3 Beyond Power Analysis and Glitching

The ChipWhisperer project currently performs power-analysis and glitch attacks on embedded computers as a method of validating security algorithms and protocols. Beyond these tasks, the ChipWhisperer can be used as a base for performing real-time detection of abnormal behaviour on embedded systems. The same power measurements being used in this work to break cryptographic algorithms can also be used to help fingerprint certain operations within an embedded computer. This type of work will become more relevant as embedded systems become increasingly complex, and they are subject to the same type of virus and worm attacks that often plague desktop computers.

It may be desired to build embedded systems that have a completely separate monitoring process (similar to a watchdog timer), rather than relying on an anti-virus or similar task running on the embedded computer itself. In this case the capture architecture presented in this thesis serves as a reference for building a very low-cost capture system which can perform the power measurement in real-time at a reasonable cost and complexity.

This can also be applied to the field of detecting counterfeit or abnormal devices. While the majority of this work has concentrated on an attacker attempting to break the security of a system, we can also consider an 'attacker' simply a supplier in a supply chain looking to increase their profit. Rather than supplying a certain grade of parts to a manufacture, they may try to substitute a lower-cost part which performs 'mostly' the same functions. Detecting such a swap may be difficult, but power analysis can be used to provide unique fingerprints that a manufacture can use to confirm a supplied part matches the expected signature.

Bibliography

[1] Agilent Technologies. *Triggering Wide-Bandwidth Sampling Oscilloscopes For Accurate Displays of High-Speed Digital Communications Waveforms*, 2005.

[2] D. Agrawal, J. Rao, and P. Rohatgi. Multi-channel Attacks. In *Proceedings of 5th Workshop on Cryptographic Hardware and Embedded Systems (CHES '03)*, volume 2779 of *Lecture Notes in Computer Science*, pages 2–16. Springer Berlin Heidelberg, 2003.

[3] M. Alderighi, F. Casini, S. d'Angelo, M. Mancini, S. Pastore, and G. Sechi. Evaluation of single event upset mitigation schemes for sram based fpgas using the flipper fault injection platform. In *Proceedings of 22nd IEEE Symposium on Defect and Fault-Tolerance in VLSI Systems (DFT)*, pages 105–113, Sept 2007.

[4] Z. Alliance. ZigBee IP Specification. Technical report, ZigBee Alliance, March 2014.

[5] R. Anderson and M. Kuhn. Tamper resistance: A cautionary note. In *Proceedings of the 2nd Conference on Proceedings of the Second USENIX Workshop on Electronic Commerce - Volume 2*, WOEC'96, pages 1–11, Berkeley, CA, USA, 1996. USENIX Association.

[6] R. Anderson and M. Kuhn. Low cost attacks on tamper resistant devices. In *Security Protocols*, pages 125–136. Springer, 1998.

[7] Atmel. Atmel AVR231: AES Bootloader. Technical report, Atmel, 2012.

[8] Atmel Corporation. ATmega48A Datasheet.

[9] Atmel Corporation. ATmega128RFA1 Datasheet, 2014.

[10] J. Balasch, B. Gierlichs, and I. Verbauwhede. An In-depth and Black-box Characterization of the Effects of Clock Glitches on 8-bit MCUs. In *Proceedings of 8th Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC '11)*, pages 105–114, 2011.

[11] J. Balasch, B. Gierlichs, R. Verdult, L. Batina, and I. Verbauwhede. Power Analysis of Atmel CryptoMemory – Recovering Keys from Secure EEPROMs. In *Proceedings of 12th International Conference on Topics in Cryptology (CT-RSA '12)*, volume 7178 of *Lecture Notes in Computer Science*, pages 19–34. Springer Berlin Heidelberg, 2012.

[12] L. Balogh. Design and application guide for high speed MOSFET gate drive circuits. Technical report, Texas Instruments/Unitrode Corporation, Power Supply Design Seminar, SEM, 2001.

[13] D. Banerjee. *PLL Performance Simulation and Design Handbook*. Texas Instruments, 4th edition, 2006.

[14] H. Bar-El, H. Choukri, D. Naccache, M. Tunstall, and C. Whelan. The Sorcerer's Apprentice Guide to Fault Attacks. *Proceedings of the IEEE*, 94(2):370–382, Feb 2006.

[15] A. Barenghi, G. Bertoni, E. Parrinello, and G. Pelosi. Low Voltage Fault Attacks on the RSA Cryptosystem. In *Proceedings of 8th Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC '09)*, pages 23–31, Sept 2009.

[16] A. Barenghi, G. Bertoni, L. Breveglieri, M. Pellicioli, and G. Pelosi. Fault attack on AES with single-bit induced faults. In *Proceedings of 6th Conference on Information Assurance and Security (IAS '10)*, pages 167–172, Aug 2010.

[17] A. Barenghi, L. Breveglieri, I. Koren, and D. Naccache. Fault Injection Attacks on Cryptographic Devices: Theory, Practice, and Countermeasures. *Proceedings of the IEEE*, 100(11):3056–3076, Nov 2012.

[18] L. Batina, J. Hogenboom, and J. van Woudenberg. Getting More from PCA: First Results of Using Principal Component Analysis for Extensive Power Analysis. In *Proceedings of 12th International Conference on Topics in Cryptology (CT-RSA '12)*, volume 7178 of *Lecture Notes in Computer Science*, pages 383–397. Springer Berlin Heidelberg, 2012.

[19] S. Belaïd, J.-S. Coron, P.-A. Fouque, B. Gérard, J.-G. Kammerer, and E. Prouff. *Improved Side-Channel Analysis of Finite-Field Multiplication*, pages 395–415. Springer Berlin Heidelberg, Berlin, Heidelberg, 2015.

[20] E. Biham and A. Shamir. Differential fault analysis of secret key cryptosystems. In *Advances in Cryptology – CRYPTO '97*, volume 1294 of *Lecture Notes in Computer Science*, pages 513–525. Springer Berlin Heidelberg, 1997.

[21] D. Boneh, R. DeMillo, and R. Lipton. On the Importance of Checking Cryptographic Protocols for Faults. In *Advances in Cryptology (EUROCRYPT '97)*, volume 1233 of *Lecture Notes in Computer Science*, pages 37–51. Springer Berlin Heidelberg, 1997.

[22] E. Brier, C. Clavier, and F. Olivier. Correlation Power Analysis with a Leakage Model. In *Proceedings of 6th Workshop on Cryptographic Hardware and Embedded Systems (CHES '04)*, volume 3156 of *Lecture Notes in Computer Science*, pages 16–29. Springer Berlin Heidelberg, 2004.

[23] E. Cagli, C. Dumas, and E. Prouff. *Enhancing Dimensionality Reduction Methods for Side-Channel Attacks*, pages 15–33. Springer International Publishing, Cham, 2016.

[24] G. Canivet, P. Maistri, R. Leveugle, J. Cldire, F. Valette, and M. Renaudin. Glitch and Laser Fault Attacks onto a Secure AES Implementation on a SRAM-Based FPGA. *Journal of Cryptology*, 24(2):247–268, 2011.

[25] G.-C. Cardarilli, F. Kaddour, A. Leandri, M. Ottavi, S. Pontarelli, and R. Velazco. Bit flip injection in processor-based architectures: a case study. In *Proceedings of 8th IEEE On-Line Testing Workshop, (IOLT '08)*, pages 117–127. IEEE, 2002.

[26] D. Carluccio. *Electromagnetic Side Channel Analysis of Embedded Crypto Devices*. PhD thesis, Ruhr-Universitat Bochum, 2005.

[27] R. Carpi, S. Picek, L. Batina, F. Menarini, D. Jakobovic, and M. Golub. Glitch It If You Can: Parameter Search Strategies for Successful Fault Injection. In *Proceedings of 12th Smart Card Research and Advanced Application Conference (CARDIS '13)*, Lecture Notes in Computer Science, pages 236–252. Springer International Publishing, 2014.

[28] I. Chadjiminas, C. Kyrkou, T. Theocharides, M. Michael, and C. Ttofis. In-field vulnerability analysis of hardware-accelerated computer vision applications. In *Proceedings of 25th Conference on Field Programmable Logic and Applications (FPL '15)*, pages 1–4, Sept 2015.

[29] T. F. Chan, G. H. Golub, and R. J. Leveque. Algorithms for Computing the Sample Variance: Analysis and Recommendations. *The American Statistician*, 37(3):242–247, 1983.

[30] S. Chari, J. Rao, and P. Rohatgi. Template Attacks. In *Proceedings of 4th Workshop on Cryptographic Hardware and Embedded Systems (CHES '02)*, volume 2523 of *Lecture Notes in Computer Science*, pages 13–28. Springer Berlin Heidelberg, 2003.

[31] O. Choudary and M. G. Kuhn. *Efficient Template Attacks*, pages 253–270. Springer International Publishing, Cham, 2014.

[32] H. Choukri and M. Tunstall. Round Reduction using Faults. In *Proceedings of 2nd Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC '05)*, pages 13–24, 2005.

[33] C. Clavier, J.-S. Coron, and N. Dabbous. Differential Power Analysis in the Presence of Hardware Countermeasures. In *Proceedings of 2nd Workshops on Cryptographic Hardware and Embedded Systems (CHES '00)*, volume 1965 of *Lecture Notes in Computer Science*, pages 252–263. Springer Berlin Heidelberg, 2000.

[34] C. Clavier, J.-L. Danger, G. Duc, M. Elaabid, B. Gêrard, S. Guilley, A. Heuser, M. Kasper, Y. Li, V. Lomné, D. Nakatsu, K. Ohta, K. Sakiyama, L. Sauvage, W. Schindler, M. Stöttinger, N. Veyrat-Charvillon, M. Walle, and A. Wurcker. Practical improvements of side-channel attacks on AES: feedback from the 2nd DPA contest. *Journal of Cryptographic Engineering*, 4(4):259–274, 2014.

[35] J. Cooper, E. DeMulder, G. Goodwill, J. Jae, G. Kenworthy, and P. Rohatgi. Test vector leakage assessment (tvla) methodology in practice. In *Proceedings of International Cryptographic Module Conference*, 2013.

[36] J. Costas. Synchronous Communications. *IRE Transactions on Communications Systems*, 5(1):99 –105, March 1957.

[37] A. Danis and B. Ors. Differential power analysis attack considering decoupling capacitance effect. In *Proceedings of 19th European Conference on Circuit Theory and Design (ECCTD '09)*, pages 359–362. IEEE, 2009.

[38] G. de Meulenaer and F.-X. Standaert. Stealthy Compromise of Wireless Sensor Nodes with Power Analysis Attacks. In *Proceedings of 2nd Conference on Mobile Lightweight Wireless Systems (MLWS '10)*, volume 45 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, pages 229–242. Springer Berlin Heidelberg, 2010.

[39] E. De Mulder. *Electromagnetic Techniques and Probes for Side-Channel Analysis on Cryptographic Devices*. PhD thesis, Katholieke Universiteit Leuven, 2010.

[40] C. Deshpande, B. Yuce, N. F. Ghalaty, D. Ganta, P. Schaumont, and L. Nazhandali. A configurable and lightweight timing monitor for fault attack detection. In *Proceedings of 2016 IEEE Computer Society Annual Symposium on VLSI (ISVLSI '16)*, pages 461–466, July 2016.

[41] A. A. Ding, C. Chen, and T. Eisenbarth. Simpler, faster, and more robust t-test based leakage detection, 2016.

[42] J. Dofe, J. Frey, and Q. Yu. Hardware security assurance in emerging iot applications. In *Proceedings of 2016 IEEE International Symposium on Circuits and Systems (ISCAS '16)*, pages 2050–2053, May 2016.

[43] J. Dofe, H. Pahlevanzadeh, and Q. Yu. A comprehensive fpga-based assessment on fault-resistant aes against correlation power analysis attack. *Journal of Electronic Testing*, 32(5):611–624, 2016.

[44] G. Duc, S. Guilley, L. Sauvage, F. Flament, M. Nassar, N. Selmane, J.-L. Danger, T. Graba, Y. Mathieu, and P. Renaud. Results of the 2009-2010 'DPA contest v2'. In *Proceedings of 2nd Workshop on Constructive Side-Channel Analysis and Secure Design (COSADE '11)*, February 2011.

[45] P. Dusart, G. Letourneux, and O. Vivolo. Differential fault analysis on A.E.S. In *Proceedings of Applied Cryptography and Network Security (ACNS '03)*, volume 2846, pages 293–306. Springer, 2003.

[46] T. Eisenbarth, T. Kasper, A. Moradi, C. Paar, M. Salmasizadeh, and M. Shalmani. On the Power of Power Analysis in the Real World: A Complete Break of the KeeLoq Code Hopping Scheme. In *Advances in Cryptology – CRYPTO 2008*, volume 5157 of *Lecture Notes in Computer Science*, pages 203–220. Springer Berlin Heidelberg, 2008.

[47] M. Faraday. Experimental researches in electricity. *Phil. Trans. R. Soc. Lond.*, 122:125–162, 1832.

[48] Freescale. MC1323x Datasheet, 2011.

[49] I. Frieslaar and B. Irwin. An investigation into the signals leakage from a smartcard based on different runtime code. In *Proceedings of Southern Africa Telecommunication Networks and Applications Conference (SATNAC '15)*, September 2015.

[50] K. Gandolfi, C. Mourtel, and F. Olivier. Electromagnetic Analysis: Concrete Results. In *Proceedings of 3rd Workshop on Cryptographic Hardware and Embedded Systems (CHES '01)*, volume 2162 of *Lecture Notes in Computer Science*, pages 251–261. Springer Berlin Heidelberg, 2001.

[51] A. Gazet. Comparative analysis of various ransomware virii. *Journal in Computer Virology*, 6(1):77–90, 2010.

[52] C. H. Gebotys, S. Ho, and C. C. Tiu. EM Analysis of Rijndael and ECC on a Wireless Java-Based PDA. In *Proceedings of 7th Workshop on Cryptographic Hardware and Embedded Systems (CHES '05)*, volume 3659 of *Lecture Notes in Computer Science*, pages 250–264. Springer, 2005.

[53] J. Geisler. Apparatus and method for precharging a bus to an intermediate level, December 30 1997. US Patent 5,703,501.

[54] G. Goodwill, B. Jun, J. Jae, and P. Rohatgi. A testing methodology for side-channel resistance validation. In *Proceedings of NIST Non-Invasive Attack Testing Workshop*, September 2011.

[55] A. Greenberg. Hackers remotely kill a jeep on the highway – with me in it. *Wired*, July 2015.

[56] S. Gueron. Intel Advanced Encryption Standard (AES) New Instructions Set. Whitepaper, 2012. Doc No. 323641-001.

[57] S. Guilley, K. Khalfallah, V. Lomne, and J.-L. Danger. Formal Framework for the Evaluation of Waveform Resynchronization Algorithms. In *Proceedings of 5th IFIP WG 11.2 Conference on Information Security Theory and Practice (WISTP '11)*, WISTP'11, pages 100–115, Berlin, Heidelberg, 2011. Springer-Verlag.

[58] H. Guntur, J. Ishii, and A. Satoh. Side-channel AttacK User Reference Architecture board SAKURA-G. In *Proceedings of 3rd Global Conference on Consumer Electronics (GCCE '14)*, pages 271–274, Oct 2014.

[59] J. Gutierrez, M. Naeve, E. Callaway, M. Bourgeois, V. Mitter, and B. Heile. IEEE 802.15.4: a developing standard for low-power low-cost wireless personal area networks. *IEEE Network*, 15(5):12–19, Sept 2001.

[60] T. Hummel. Exploring Effects of Electromagnetic Fault Injection on a 32-bit High Speed Embedded Device Microprocessor. Master's thesis, University of Twente, July 2014.

[61] D.-G. H. Hyunjin Ahn. Multilateral white-box cryptanalysis: Case study on wb-aes of ches challenge 2016. Cryptology ePrint Archive, Report 2016/807, 2016. `http://eprint.iacr.org/2016/807`.

[62] IEEE. Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs). *IEEE Std. 802.15.4-2006*, 2006.

[63] M. G. Institute. The internet of things: Mapping the value beyond the hype. *McKinsey Global Institute*, June 2015.

[64] J. Jaffe. A First-Order DPA Attack Against AES in Counter Mode with Unknown Initial Counter. In *Proceedings of 8th Workshop on Cryptographic Hardware and Embedded Systems (CHES '06)*, volume 4727 of *Lecture Notes in Computer Science*, pages 1–13. Springer Berlin Heidelberg, 2007.

[65] B. Jun and G. Kenworthy. Is your mobile device radiating keys? In *RSA Conference 2012*, 2012.

[66] M. Kafi, S. Guilley, S. Marcello, and D. Naccache. Deconvolving Protected Signals. In *Proceedings of 10th Conference on Availability, Reliability and Security (ARES '09).*, pages 687 –694, March 2009.

[67] T. Kasper, D. Oswald, and C. Paar. A Versatile Framework for Implementation Attacks on Cryptographic RFIDs and Embedded Devices. In *Transactions on Computational Science X*, pages 100–130, Berlin, Heidelberg, 2010. Springer-Verlag.

[68] F. Kastensmidt, L. Carro, and R. Reis. *Fault-Tolerance Techniques for SRAM-based FPGAs*, volume 32. Springer, 2006.

[69] T. Katashita, A. Satoh, K. Kikuchi, H. Nakagawa, and M. Aoyagi. Evaluation of dpa characteristics of sasebo for board level simulations. In *Proceedings of 1st Workshop on Constructive Side-Channel Analysis and Secure Design (COSADE '10)*, pages 36–39, 2010.

[70] F. Khelil, M. Hamdi, S. Guilley, J. Danger, and N. Selmane. Fault analysis attack on an fpga aes implementation. In *Proceedings of 2nd International Conference on New Technologies, Mobility and Security (NTMS '08).*, pages 1–5, Nov 2008.

[71] I. Kizhvatov. Side Channel Analysis of AVR XMEGA Crypto Engine. In *Proceedings of 4th Workshop on Embedded Systems Security (WESS '09)*, WESS '09, pages 8:1–8:7, New York, NY, USA, 2009. ACM.

[72] P. Kocher, J. Jaffe, and B. Jun. Differential power analysis. In *Advances in Cryptology - CRYPTO' 99*, pages 388–397. Springer-Verlag, 1999.

[73] P. C. Kocher. Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. In *Advances in Cryptology – CRYPTO 1996*, CRYPTO '96, pages 104–113, London, UK, UK, 1996. Springer-Verlag.

[74] F. Koeune, F. Koeune, J.-J. Quisquater, and J. jacques Quisquater. A timing attack against Rijndael. Technical report, UCL Crypto Group, 1999.

[75] T. Korak and M. Hoefle. On the Effects of Clock and Power Supply Tampering on Two Microcontroller Platforms. In *Proceedings of 11th Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC '14)*, September 2014.

[76] A. C. Kunming. Comparison of Side Channel Analysis Measurement Setups. Master's thesis, Technische Universiteit Eindhoven, 2015.

[77] J. P. Lewis. Fast Template Matching. In *Proceedings of 8th Canadian Conference on Vision Interface (VI '95)*, pages 120–123, 1995.

[78] L. Li, J. Kim, H. Wang, S. Wu, Y. Takita, H. Takeuchi, K. Araki, and J. Fan. Measurement of multiple switching current components through a bulk decoupling capacitor using a lab-made low-cost current probe. In *Proceedings of 8th Symposium on Electromagnetic Compatibility (EMC '11)*, pages 417–421. IEEE, 2011.

[79] H. Maghrebi, T. Portigliatti, and E. Prouff. Breaking cryptographic implementations using deep learning techniques. Cryptology ePrint Archive, Report 2016/921, 2016. `http://eprint.iacr.org/2016/921`.

[80] S. Mangard, E. Oswald, and T. Popp. *Power Analysis Attacks: Revealing the Secrets of Smart Cards*. Advances in information security. Springer, 2008.

[81] J. Massey. Guessing and entropy. In *Proceedings of IEEE International Symposium on Information Theory (ISIT '94)*, page 204, 1994.

[82] E. Mateos and C. Gebotys. Side channel analysis using giant magneto-resistive (gmr) sensors. In *Proceedings of 2nd Workshop on Constructive Side-Channel Analysis and Secure Design (COSADE '11)*, 2011.

[83] C. McGoogan. Smart feeder outage left pets hungry for 10 hours. *The Telegraph*, July 2016.

[84] C. Mead and L. Conway. *Introduction to VLSI systems*. Addison-Wesley, 1980.

[85] T. Messerges. *Power Analysis Attacks and Countermeasures for Cryptographic Algorithms*. PhD thesis, University of Illinois at Chicago, 2000.

[86] Microchip. Microchip technology launches industrys first low-cost 18-pin EPROM-based microcontrollers. Apr 1996.

[87] C. Miller and C. Valasek. Remote exploitation of an unaltered passenger vehicle. In *Proceedings of Black Hat 2015*, August 2015.

[88] D. Montminy, R. Baldwin, M. Temple, and E. Laspe. Improving cross-device attacks using zero-mean unit-variance normalization. *Journal of Cryptographic Engineering*, 3(2):99–110, 2013.

[89] A. Moradi, O. Mischke, and T. Eisenbarth. Correlation-enhanced power analysis collision attack. In *Proceedings of 12th Workshop on Cryptographic Hardware and Embedded Systems (CHES '10)*, pages 125–139. Springer-Verlag, 2011.

[90] A. Moradi, A. Barenghi, T. Kasper, and C. Paar. On the Vulnerability of FPGA Bitstream Encryption Against Power Analysis Attacks: Extracting Keys from Xilinx Virtex-II FPGAs. In *Proceedings of 18th ACM Conference on Computer and Communications Security (CCS '11)*, pages 111–124. ACM, 2011.

[91] A. Moradi and G. Hinterwälder. *Side-Channel Security Analysis of Ultra-Low-Power FRAM-Based MCUs*, pages 239–254. Springer International Publishing, Cham, 2015.

[92] A. Moradi, M. Kasper, and C. Paar. On the Portability of Side-Channel Attacks – An Analysis of the Xilinx Virtex 4, Virtex 5, and Spartan 6 Bitstream Encryption Mechanism. Cryptology ePrint Archive, Report 2011/391, 2011. `http://eprint.iacr.org/`.

[93] A. Moradi, M. Kasper, and C. Paar. Black-Box Side-Channel Attacks Highlight the Importance of Countermeasures. In *Proceedings of 12th International Conference on Topics in Cryptology (CT-RSA '12)*, volume 7178 of *Lecture Notes in Computer Science*, pages 1–18. Springer Berlin Heidelberg, 2012.

[94] A. Moradi, D. Oswald, C. Paar, and P. Swierczynski. Side-channel Attacks on the Bitstream Encryption Mechanism of Altera Stratix II: Facilitating Black-box Analysis Using Software Reverse-engineering. In *Proceedings of 21st*

ACM/SIGDA Symposium on Field Programmable Gate Arrays (FPGA '13), pages 91–100. ACM, 2013.

[95] N. Moro, A. Dehbaoui, K. Heydemann, B. Robisson, and E. Encrenaz. Electromagnetic fault injection: Towards a fault model on a 32-bit microcontroller. In *Proceedings of 10th Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC '13)*, pages 77–88, Aug 2013.

[96] E. Nascimento, L. Chmielewski, D. Oswald, and P. Schwabe. Attacking embedded ecc implementations through cmov side channels. In *Proceedings of Selected Areas in Cryptography (SAC '16)*, 2016. `http://eprint.iacr.org/2016/923`.

[97] E. Nascimento, J. López, and R. Dahab. *Efficient and Secure Elliptic Curve Cryptography for 8-bit AVR Microcontrollers*, pages 289–309. Springer International Publishing, Cham, 2015.

[98] M. Neve and K. Tiri. On the complexity of side-channel attacks on aes-256 – methodology and quantitative results on cache attacks. Cryptology ePrint Archive, Report 2007/318, 2007. http://eprint.iacr.org/.

[99] D. North. An analysis of the factors which determine signal/noise discrimination in pulsed-carrier systems. *RCA Labs.*, 1943.

[100] C. O'Flynn and Z. D. Chen. Side channel power analysis of an aes-256 bootloader. In *2015 IEEE 28th Canadian Conference on Electrical and Computer Engineering (CCECE)*, pages 750–755, May 2015.

[101] C. O'Flynn and C. Zhizhang. A case study of Side-Channel Analysis using Decoupling Capacitor Power Measurement with the OpenADC. In *Proceedings of Workshop on Foundations and Practices of Security (FPS '13)*, volume 7743, pages 328–344. Springer, 2013.

[102] C. O'Flynn. A lightbulb worm? In *Proceedings of Black Hat 2016*, August 2016.

[103] C. O'Flynn and Z. Chen. Synchronous sampling and clock recovery of internal oscillators for side channel analysis and fault injection. *Journal of Cryptographic Engineering*, 5(1):53–69, 2015.

[104] C. O'Flynn and Z. Chen. *Power Analysis Attacks Against IEEE 802.15.4 Nodes*, pages 55–70. Springer International Publishing, Cham, 2016.

[105] C. O'Flynn and Z. Chen. ChipWhisperer: An Open-Source Platform for Hardware Embedded Security Research. In *Proceedings of 5th Workshop on Constructive Side-Channel Analysis and Secure Design (COSADE '14)*, volume 8622 of *Lecture Notes in Computer Science*, pages 243–260. Springer International Publishing, 2014.

[106] J. Olivares, J. Hormigo, J. Villalba, and I. Benavides. Minimum Sum of Absolute Differences Implementation in a Single FPGA Device. In *Proceedings of 14th Conference on Field Programmable Logic and Application (FPL '04)*, volume 3203 of *Lecture Notes in Computer Science*, pages 986–990. Springer Berlin Heidelberg, 2004.

[107] D. Oswald. *Implementation Attacks: From Theory to Practice*. PhD thesis, Ruhr-Universitat Bochum, 2013.

[108] D. Oswald, T. Kasper, S. Markhoff, and C. Paar. FPGA-based Implementation Attacks with GIAnT, November 2011. 9th CrypArchi Workschop, Bochum.

[109] D. Oswald and C. Paar. Breaking Mifare DESFire MF3ICD40: Power Analysis and Templates in the Real World. In *Proceedings of 13th Workshop on Cryptographic Hardware and Embedded Systems (CHES '11)*, volume 6917 of *Lecture Notes in Computer Science*, pages 207–222. Springer Berlin Heidelberg, 2011.

[110] D. Oswald, B. Richter, and C. Paar. Side-Channel Attacks on the Yubikey 2 One-Time Password Generator. In *Proceedings of 16th Symposium on Research in Attacks, Intrusions, and Defenses (RAID '13)*, volume 8145 of *Lecture Notes in Computer Science*, pages 204–222. Springer Berlin Heidelberg, 2013.

[111] D. Oswald, D. Strobel, F. Schellenberg, T. Kasper, and C. Paar. When Reverse-Engineering Meets Side-Channel Analysis – Digital Lockpicking in Practice. In *Proceedings of 20th Conference on Selected Areas in Cryptography (SAC '13)*, volume 8282 of *Lecture Notes in Computer Science*, pages 571–588. Springer Berlin Heidelberg, 2014.

[112] E. Oswald. OpenSCA: A Matlab-based open source framework for side-channel attacks, 2009. `http://opensca.sourceforge.net/`.

[113] H. Pahlevanzadeh, J. Dofe, and Q. Yu. Assessing cpa resistance of aes with different fault tolerance mechanisms. In *Proceedings of 2016 21st Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 661–666, Jan 2016.

[114] J.-J. Quisquater and D. Samyde. Eddy current for Magnetic Analysis with Active Sensor. In *Proceedings of Esmart Conference '02*, 9 2002.

[115] D. Réal, C. Canovas, J. Clédière, M. Drissi, and F. Valette. Defeating Classical Hardware Countermeasures: A New Processing for Side Channel Analysis. In *Proceedings of 11th Conference on Design, Automation and Test in Europe (DATE '08)*, DATE '08, pages 1274–1279, New York, NY, USA, 2008. ACM.

[116] R. Rodrigues, S. Kundu, and O. Khan. Shadow checker (sc): A low-cost hardware scheme for online detection of faults in small memory structures of a microprocessor. In *Proceedings of 41st IEEE International Test Conference (ITC '10)*, pages 1–10, Nov 2010.

[117] E. Ronen, C. OFlynn, A. Shamir, and A.-O. Weingarten. Iot goes nuclear: Creating a zigbee chain reaction. Cryptology ePrint Archive, Report 2016/1047, 2016. http://eprint.iacr.org/2016/1047.

[118] N. Sastry and D. Wagner. Security Considerations for IEEE 802.15.4 Networks. In *Proceedings of 3rd ACM Workshop on Wireless Security (WiSe '04)*, pages 32–42. ACM, 2004.

[119] A. Satoh. Side-channel attack standard evaluation board (sasebo). http://www.morita-tech.co.jp/SASEBO/en/index.html, 2011.

[120] J. Schmidt and C. Herbst. A practical fault attack on square and multiply. In *Proceedings of 5th Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC '08)*, pages 53–58, Aug 2008.

[121] M. Scott. scanlime:015 / Glitchy Descriptor Firmware Grab, October 2016. https://www.youtube.com/watch?v=TeCQatNcF20.

[122] Silicon Labratories. E35x Datasheet, 2013.

[123] S. Skorobogatov. Synchronization method for SCA and fault attacks. *Journal of Cryptographic Engineering*, 1(1):71–77, April 2011.

[124] S. Skorobogatov and R. Anderson. Optical fault induction attacks. In *Proceedings of 4th Workshop on Cryptographic Hardware and Embedded Systems (CHES '02)*, volume 2523 of *Lecture Notes in Computer Science*, pages 2–12. Springer Berlin Heidelberg, 2003.

[125] S. Skorobogatov and C. Woods. Breakthrough silicon scanning discovers backdoor in military chip. In *Proceedings of 14th Workshop on Cryptographic Hardware and Embedded Systems (CHES '12)*, volume 7428 of *Lecture Notes in Computer Science*, pages 23–40. Springer Berlin Heidelberg, 2012.

[126] D. Smith. Signal and noise measurement techniques using magnetic field probes. In *Electromagnetic Compatibility, 1999 IEEE International Symposium on*, volume 1, pages 559–563. IEEE, 1999.

[127] Y. Souissi, J. Danger, S. Guilley, S. Bhasin, and M. Nassar. Embedded systems security: An evaluation methodology against side channel attacks. In *Proceedings of 5th Conference on Design and Architectures for Signal and Image Processing (DASIP '11)*, pages 1–8. IEEE, 2011.

[128] F.-X. Standaert, T. Malkin, and M. Yung. A Unified Framework for the Analysis of Side-Channel Key Recovery Attacks. In *Advances in Cryptology - (EUROCRYPT '09)*, volume 5479 of *Lecture Notes in Computer Science*, pages 443–461. Springer Berlin Heidelberg, 2009.

[129] F. Standaert and C. Archambeau. Using subspace-based template attacks to compare and combine power and electromagnetic information leakages. In *Proceedings of 10th Workshop on Cryptographic Hardware and Embedded Systems (CHES '08)*, pages 411–425. Springer-Verlag, 2008.

[130] STMicroelectronics. STM32W108xx Datasheet, 2013.

[131] I. Studnia, V. Nicomette, E. Alata, Y. Deswarte, M. Kaniche, and Y. Laarouchi. Survey on security threats and protection mechanisms in embedded automotive networks. In *Proceedings of 43rd Annual IEEE/IFIP Conference on Dependable Systems and Networks Workshop (DSN-W '13)*, pages 1–12, June 2013.

[132] T. Sugawara, D. Suzuki, M. Saeki, M. Shiozaki, and T. Fujino. On measurable side-channel leaks inside asic design primitives. *Journal of Cryptographic Engineering*, 4(1):59–73, 2014.

[133] P. Swierczynski, A. Moradi, D. Oswald, and C. Paar. Physical Security Evaluation of the Bitstream Encryption Mechanism of Altera Stratix II and Stratix III FPGAs. *ACM Transactions on Reconfigurable Technology and Systems*, 7(4):7:1–7:23, December 2014.

[134] Texas Instruments. CC2530F Datasheet, 2015.

[135] Q. Tian and S. Huss. On Clock Frequency Effects in Side Channel Attacks of Symmetric Block Ciphers. In *Proceedings of 5th Conference on New Technologies, Mobility and Security (NTMS '12)*, pages 1 –5, May 2012.

[136] J. G. J. van Woudenberg, M. F. Witteman, and B. Bakker. Improving Differential Power Analysis by Elastic Alignment. In *Proceedings of 11th International Conference on Topics in Cryptology (CT-RSA '11)*, CT-RSA'11, pages 104–119, Berlin, Heidelberg, 2011. Springer-Verlag.

[137] J. van Woudenberg, M. Witteman, and F. Menarini. Practical Optical Fault Injection on Secure Microcontrollers. In *Proceedings of 8th Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC '11)*, pages 91–99, Sept 2011.

[138] R. Velegalati and J.-P. Kaps. Introducing FOBOS: Flexible Open-source BOard for Side-channel analysis. Work in Progress (WiP), Third International Workshop on Constructive Side-Channel Analysis and Secure Design, COSADE 2012, May 2012.

[139] R. Velegalati and J.-P. Kaps. Towards a Flexible, Opensource BOard for Side-channel analysis (FOBOS). Cryptographic architectures embedded in reconfigurable devices, CRYPTARCHI 2013, June 2013.

[140] N. Veyrat-Charvillon, B. Grard, M. Renauld, and F.-X. Standaert. An Optimal Key Enumeration Algorithm and Its Application to Side-Channel Attacks. In *Proceedings of 20th Conference on Selected Areas in Cryptography (SAC '13)*,

volume 7707 of *Lecture Notes in Computer Science*, pages 390–406. Springer Berlin Heidelberg, 2013.

[141] M. Violante, L. Sterpone, A. Manuzzato, S. Gerardin, P. Rech, M. Bagatin, A. Paccagnella, C. Andreani, G. Gorini, A. Pietropaolo, et al. A new hardware/-software platform and a new 1/e neutron source for soft error studies: Testing fpgas at the isis facility. *IEEE Transactions on Nuclear Science*, 54(4):1184–1189, 2007.

[142] J. Wang, R. Katz, J. Sun, B. Cronquist, J. McCollum, T. Speers, and W. Plants. Sram based re-programmable fpga for space applications. *IEEE Transactions on Nuclear Science*, 46(6):1728–1735, Dec 1999.

[143] J. Weaver and M. Horowitz. Measurement of via currents in printed circuit boards using inductive loops. In *Proceedings of 15th Conference on Electrical Performance of Electronic Packaging (EPEP '06)*, pages 37–40. IEEE, 2006.

[144] J. Weaver and M. Horowitz. Measurement of supply pin current distributions in integrated circuit packages. In *Proceedings of 16th Conference on Electrical Performance of Electronic Packaging (EPEP '07)*, pages 7–10. IEEE, 2007.

[145] D. Whiting, N. Ferguson, and R. Housley. Counter with CBC-MAC (CCM). `https://tools.ietf.org/html/rfc3610`.

[146] Xilinx. Spartan-6 fpga configuration user guide (ug380). Technical report, Xilinx, 2015.

[147] S. Yang, P. Gupta, M. Wolf, D. Serpanos, V. Narayanan, and Y. Xie. Power analysis attack resistance engineering by dynamic voltage and frequency scaling. *ACM Transactions on Embedded Computer Systems*, 11(3):62:1–62:16, September 2012.

[148] B. Yuce, N. F. Ghalaty, and P. Schaumont. Improving fault attacks on embedded software using risc pipeline characterization. In *Proceedings of Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC '15)*, pages 97–108, Sept 2015.

[149] L. Zussa, J.-M. Dutertre, J. Clediere, and B. Robisson. Analysis of the fault injection mechanism related to negative and positive power supply glitches using an on-chip voltmeter. In *Proceedings of 7th IEEE International Symposium on Hardware-Oriented Security and Trust (HOST '14)*, pages 130–135, May 2014.

Appendix A

Publications and Contributions

The following briefly summarizes my contributions in this field which I accomplished during the PhD.

A.1 Peer Reviewed Publications

- Eyal Ronen, Colin OFlynn, Adi Shamir and Achi-Or Weingarten. IoT Goes Nuclear: Creating a ZigBee Chain Reaction. In *2017 IEEE Symposium on Security and Privacy (SP)*, 2017. [14% acceptance rate]
- Colin O'Flynn and Zhizhang Chen. Power Analysis Attacks against IEEE 802.15.4 Nodes. In *Proceedings of 7th Workshop on Constructive Side-Channel Analysis and Secure Design (COSADE '16)*, 2016. [37% acceptance rate]
- Colin O'Flynn and Zhizhang Chen. Side channel power analysis of an AES-256 Bootloader. In *Proceedings of IEEE 28th Canadian Conference on Electrical and Computer Engineering (CCECE '15)*, 2015.
- Colin O'Flynn and Zhizhang Chen. Synchronous sampling and clock recovery of internal oscillators for side channel analysis and fault injection. In *Journal of Cryptographic Engineering*, volume 5, pages 53 – 69. Springer-Verlag, 2015.
- Colin O'Flynn and Zhizhang Chen. ChipWhisperer: An Open-Source Platform for Hardware Embedded Security Research. In *Proceedings of 5th Workshop on Constructive Side-Channel Analysis and Secure Design (COSADE '14)*, pages 243–260. 2014. [39% acceptance rate]
- Colin O'Flynn and Zhizhang Chen. A case study of Side-Channel Analysis using Decoupling Capacitor Power Measurement with the OpenADC. In emph-Proceedings of Workshop on Foundations and Practices of Security (FPS '13), pages 328–344. 2013. [35% acceptance rate]

Most of these publications centered around the open-source ChipWhisperer project. This project has already begun to impact the field; the following list papers where the authors use the ChipWhisperer hardware themselves (i.e., I've excluded those simply referencing my publication), and where I have not worked with the authors directly myself: [19, 23, 40, 42, 43, 49, 61, 76, 79, 96, 97, 113, 148].

A.2 Academic Conference Activities

In addition to publishing in academic conferences, I participated in additional ways for two of the CHES conferences. At CHES 2013 I presented a half-day tutorial on the use of ChipWhisperer and setup of a low-cost analysis lab.

At CHES 2016 I was responsible for running a Capture the Flag (CTF) event. This event saw researchers from around the world submitting various encryption standards, which were run on a physical embedded system in my office and power traces collected. The power traces were uploaded to a web server, and other researchers then attempted to perform power analysis using these traces.

A.3 Industry Conference Presentations

As part of helping to spread this knowledge to industry, I participated in a number of industry-focused conferences. These conferences typically had some form of selection based on the speaker and topic, rather than an academic paper. While many of the presentations involved significant white papers released at the same time, the paper itself was not subject to peer review. The following lists the conference presentations related to this thesis:

- Black Hat Abu Dhabi 2012 - Power Analysis for Cheapskates
- Black Hat EU 2013 - Power Analysis for Cheapskates
- Black Hat USA 2013 - Power Analysis Attacks for Cheapskates
- AtlSecCon 2014 - Hacking Embedded Systems
- RECON 2014 - Power Analysis and Clock Glitching with ChipWhisperer

- Black Hat USA 2014 - 2-Day Training ('Advanced Hardware Hacking: Hands-on Power Analysis and Glitching with the ChipWhisperer')
- AtlSecCon 2015 - In Hardware We Trust
- RECON 2015 - Glitching and Side Channel Analysis for All
- DEFCON 2015 - Power Analysis Talk
- AtlSecCon 2016 - Hardware Hacking - Lightbulbs and Hard Drives
- Black Hat USA 2016 - A Lightbulb Worm?
- Black Hat USA 2016 - Brute-Forcing Lockdown Harddrive PIN Codes
- Black Hat USA 2016 - 2-Day Training ('Advanced Harddware Hacking: Hands-on Power Analysis and Glitching with the ChipWhisperer')
- VOLPE Open Car Hacking Event 2016 - ChipWhisperer for SCA and Fault Injection
- Intel Security Conference 2016 - IoT - A Lightbulb Worm?

A.4 Industry Publications

During this thesis, I also wrote several articles for Circuit Cellar Ink magazine. The articles of relevance to this thesis included:

- Colin O'Flynn. Partial Reconfiguration. *Circuit Cellar*, June 2014.
- Colin O'Flynn. USB-to-FPGA Communication. *Circuit Cellar*, June 2015.
- Colin O'Flynn. Breaking Unbreakable Cryptography with Power Analysis Attacks. *Circuit Cellar*, August 2016.
- Colin O'Flynn. Experiment with Gitch Attacks on FPGAs. *Circuit Cellar*, December 2016.
- Colin O'Flynn. Timing and Power Attacks. *Circuit Cellar*, April 2017.

- Colin O'Flynn. Breaking a Password with Power Analysis Attacks. *Circuit Cellar*, June 2017.

Appendix B

Copyright Permission Letters

B.1 Side channel power analysis of an AES-256 Bootloader [100]

B.2 Power Analysis Attacks against IEEE 802.15.4 Nodes [104]

SPRINGER LICENSE
TERMS AND CONDITIONS

Mar 24, 2017

This Agreement between Colin O'Flynn ("You") and Springer ("Springer") consists of your license details and the terms and conditions provided by Springer and Copyright Clearance Center.

License Number	4075590426762
License date	Mar 24, 2017
Licensed Content Publisher	Springer
Licensed Content Publication	Springer eBook
Licensed Content Title	Power Analysis Attacks Against IEEE 802.15.4 Nodes
Licensed Content Author	Colin O'Flynn
Licensed Content Date	Jan 1, 2016
Type of Use	Thesis/Dissertation
Portion	Full text
Number of copies	1
Author of this Springer article	Yes and you are the sole author of the new work
Order reference number	
Title of your thesis / dissertation	A Framework for Embedded Hardware Security Analysis
Expected completion date	Apr 2017
Estimated size(pages)	195
Requestor Location	Colin O'Flynn 1083 Queen St., Suite 196 Halifax, NS B3H0B2 Canada Attn: Colin O'Flynn
Billing Type	Invoice
Billing Address	Colin O'Flynn 1083 Queen St., Suite 196 Halifax, NS B3H0B2 Canada Attn: Colin O'Flynn
Total	0.00 USD
Terms and Conditions	

Introduction
The publisher for this copyrighted material is Springer. By clicking "accept" in connection with completing this licensing transaction, you agree that the following terms and conditions apply to this transaction (along with the Billing and Payment terms and conditions

established by Copyright Clearance Center, Inc. ("CCC"), at the time that you opened your Rightslink account and that are available at any time at http://myaccount.copyright.com).

Limited License

With reference to your request to reuse material on which Springer controls the copyright, permission is granted for the use indicated in your enquiry under the following conditions:

- Licenses are for one-time use only with a maximum distribution equal to the number stated in your request.
- Springer material represents original material which does not carry references to other sources. If the material in question appears with a credit to another source, this permission is not valid and authorization has to be obtained from the original copyright holder.
- This permission
• is non-exclusive
• is only valid if no personal rights, trademarks, or competitive products are infringed.
• explicitly excludes the right for derivatives.
- Springer does not supply original artwork or content.
- According to the format which you have selected, the following conditions apply accordingly:
• **Print and Electronic:** This License include use in electronic form provided it is password protected, on intranet, or CD-Rom/DVD or E-book/E-journal. It may not be republished in electronic open access.
• **Print:** This License excludes use in electronic form.
• **Electronic:** This License only pertains to use in electronic form provided it is password protected, on intranet, or CD-Rom/DVD or E-book/E-journal. It may not be republished in electronic open access.

For any electronic use not mentioned, please contact Springer at permissions.springer@spi-global.com.

- Although Springer controls the copyright to the material and is entitled to negotiate on rights, this license is only valid subject to courtesy information to the author (address is given in the article/chapter).
- If you are an STM Signatory or your work will be published by an STM Signatory and you are requesting to reuse figures/tables/illustrations or single text extracts, permission is granted according to STM Permissions Guidelines: http://www.stm-assoc.org/permissions-guidelines/

For any electronic use not mentioned in the Guidelines, please contact Springer at permissions.springer@spi-global.com. If you request to reuse more content than stipulated in the STM Permissions Guidelines, you will be charged a permission fee for the excess content.

Permission is valid upon payment of the fee as indicated in the licensing process. If permission is granted free of charge on this occasion, that does not prejudice any rights we might have to charge for reproduction of our copyrighted material in the future.

-If your request is for reuse in a Thesis, permission is granted free of charge under the following conditions:

This license is valid for one-time use only for the purpose of defending your thesis and with a maximum of 100 extra copies in paper. If the thesis is going to be published, permission needs to be reobtained.

- includes use in an electronic form, provided it is an author-created version of the thesis on his/her own website and his/her university's repository, including UMI (according to the definition on the Sherpa website: http://www.sherpa.ac.uk/romeo/);
- is subject to courtesy information to the co-author or corresponding author.

Geographic Rights: Scope

Licenses may be exercised anywhere in the world.

Altering/Modifying Material: Not Permitted

Figures, tables, and illustrations may be altered minimally to serve your work. You may not

alter or modify text in any manner. Abbreviations, additions, deletions and/or any other alterations shall be made only with prior written authorization of the author(s).

Reservation of Rights

Springer reserves all rights not specifically granted in the combination of (i) the license details provided by you and accepted in the course of this licensing transaction and (ii) these terms and conditions and (iii) CCC's Billing and Payment terms and conditions.

License Contingent on Payment

While you may exercise the rights licensed immediately upon issuance of the license at the end of the licensing process for the transaction, provided that you have disclosed complete and accurate details of your proposed use, no license is finally effective unless and until full payment is received from you (either by Springer or by CCC) as provided in CCC's Billing and Payment terms and conditions. If full payment is not received by the date due, then any license preliminarily granted shall be deemed automatically revoked and shall be void as if never granted. Further, in the event that you breach any of these terms and conditions or any of CCC's Billing and Payment terms and conditions, the license is automatically revoked and shall be void as if never granted. Use of materials as described in a revoked license, as well as any use of the materials beyond the scope of an unrevoked license, may constitute copyright infringement and Springer reserves the right to take any and all action to protect its copyright in the materials.

Copyright Notice: Disclaimer

You must include the following copyright and permission notice in connection with any reproduction of the licensed material:

"Springer book/journal title, chapter/article title, volume, year of publication, page, name(s) of author(s), (original copyright notice as given in the publication in which the material was originally published) "With permission of Springer"

In case of use of a graph or illustration, the caption of the graph or illustration must be included, as it is indicated in the original publication.

Warranties: None

Springer makes no representations or warranties with respect to the licensed material and adopts on its own behalf the limitations and disclaimers established by CCC on its behalf in its Billing and Payment terms and conditions for this licensing transaction.

Indemnity

You hereby indemnify and agree to hold harmless Springer and CCC, and their respective officers, directors, employees and agents, from and against any and all claims arising out of your use of the licensed material other than as specifically authorized pursuant to this license.

No Transfer of License

This license is personal to you and may not be sublicensed, assigned, or transferred by you without Springer's written permission.

No Amendment Except in Writing

This license may not be amended except in a writing signed by both parties (or, in the case of Springer, by CCC on Springer's behalf).

Objection to Contrary Terms

Springer hereby objects to any terms contained in any purchase order, acknowledgment, check endorsement or other writing prepared by you, which terms are inconsistent with these terms and conditions or CCC's Billing and Payment terms and conditions. These terms and conditions, together with CCC's Billing and Payment terms and conditions (which are incorporated herein), comprise the entire agreement between you and Springer (and CCC) concerning this licensing transaction. In the event of any conflict between your obligations established by these terms and conditions and those established by CCC's Billing and Payment terms and conditions, these terms and conditions shall control.

Jurisdiction

All disputes that may arise in connection with this present License, or the breach thereof,

shall be settled exclusively by arbitration, to be held in the Federal Republic of Germany, in accordance with German law.

Other conditions:

V 12AUG2015

B.3 Synchronous sampling and clock recovery of internal oscillators for side channel analysis and fault injection [103]

SPRINGER LICENSE
TERMS AND CONDITIONS

Mar 24, 2017

This Agreement between Colin O'Flynn ("You") and Springer ("Springer") consists of your license details and the terms and conditions provided by Springer and Copyright Clearance Center.

License Number	4075590518653
License date	Mar 24, 2017
Licensed Content Publisher	Springer
Licensed Content Publication	Journal of Cryptographic Engineering
Licensed Content Title	Synchronous sampling and clock recovery of internal oscillators for side channel analysis and fault injection
Licensed Content Author	Colin O'Flynn
Licensed Content Date	Jan 1, 2014
Licensed Content Volume	5
Licensed Content Issue	1
Type of Use	Thesis/Dissertation
Portion	Full text
Number of copies	1
Author of this Springer article	Yes and you are the sole author of the new work
Order reference number	
Title of your thesis / dissertation	A Framework for Embedded Hardware Security Analysis
Expected completion date	Apr 2017
Estimated size(pages)	195
Requestor Location	Colin O'Flynn 1083 Queen St., Suite 196 Halifax, NS B3H0B2 Canada Attn: Colin O'Flynn
Billing Type	Invoice
Billing Address	Colin O'Flynn 1083 Queen St., Suite 196 Halifax, NS B3H0B2 Canada Attn: Colin O'Flynn
Total	0.00 USD
Terms and Conditions	

Introduction
The publisher for this copyrighted material is Springer. By clicking "accept" in connection

with completing this licensing transaction, you agree that the following terms and conditions apply to this transaction (along with the Billing and Payment terms and conditions established by Copyright Clearance Center, Inc. ("CCC"), at the time that you opened your Rightslink account and that are available at any time at http://myaccount.copyright.com).

Limited License

With reference to your request to reuse material on which Springer controls the copyright, permission is granted for the use indicated in your enquiry under the following conditions:

- Licenses are for one-time use only with a maximum distribution equal to the number stated in your request.
- Springer material represents original material which does not carry references to other sources. If the material in question appears with a credit to another source, this permission is not valid and authorization has to be obtained from the original copyright holder.
- This permission
• is non-exclusive
• is only valid if no personal rights, trademarks, or competitive products are infringed.
• explicitly excludes the right for derivatives.
- Springer does not supply original artwork or content.
- According to the format which you have selected, the following conditions apply accordingly:
• **Print and Electronic:** This License include use in electronic form provided it is password protected, on intranet, or CD-Rom/DVD or E-book/E-journal. It may not be republished in electronic open access.
• **Print:** This License excludes use in electronic form.
• **Electronic:** This License only pertains to use in electronic form provided it is password protected, on intranet, or CD-Rom/DVD or E-book/E-journal. It may not be republished in electronic open access.

For any electronic use not mentioned, please contact Springer at permissions.springer@spi-global.com.

- Although Springer controls the copyright to the material and is entitled to negotiate on rights, this license is only valid subject to courtesy information to the author (address is given in the article/chapter).
- If you are an STM Signatory or your work will be published by an STM Signatory and you are requesting to reuse figures/tables/illustrations or single text extracts, permission is granted according to STM Permissions Guidelines: http://www.stm-assoc.org/permissions-guidelines/

For any electronic use not mentioned in the Guidelines, please contact Springer at permissions.springer@spi-global.com. If you request to reuse more content than stipulated in the STM Permissions Guidelines, you will be charged a permission fee for the excess content.

Permission is valid upon payment of the fee as indicated in the licensing process. If permission is granted free of charge on this occasion, that does not prejudice any rights we might have to charge for reproduction of our copyrighted material in the future.

-If your request is for reuse in a Thesis, permission is granted free of charge under the following conditions:

This license is valid for one-time use only for the purpose of defending your thesis and with a maximum of 100 extra copies in paper. If the thesis is going to be published, permission needs to be reobtained.

- includes use in an electronic form, provided it is an author-created version of the thesis on his/her own website and his/her university's repository, including UMI (according to the definition on the Sherpa website: http://www.sherpa.ac.uk/romeo/);
- is subject to courtesy information to the co-author or corresponding author.

Geographic Rights: Scope

Licenses may be exercised anywhere in the world.

Altering/Modifying Material: Not Permitted

Figures, tables, and illustrations may be altered minimally to serve your work. You may not alter or modify text in any manner. Abbreviations, additions, deletions and/or any other alterations shall be made only with prior written authorization of the author(s).

Reservation of Rights

Springer reserves all rights not specifically granted in the combination of (i) the license details provided by you and accepted in the course of this licensing transaction and (ii) these terms and conditions and (iii) CCC's Billing and Payment terms and conditions.

License Contingent on Payment

While you may exercise the rights licensed immediately upon issuance of the license at the end of the licensing process for the transaction, provided that you have disclosed complete and accurate details of your proposed use, no license is finally effective unless and until full payment is received from you (either by Springer or by CCC) as provided in CCC's Billing and Payment terms and conditions. If full payment is not received by the date due, then any license preliminarily granted shall be deemed automatically revoked and shall be void as if never granted. Further, in the event that you breach any of these terms and conditions or any of CCC's Billing and Payment terms and conditions, the license is automatically revoked and shall be void as if never granted. Use of materials as described in a revoked license, as well as any use of the materials beyond the scope of an unrevoked license, may constitute copyright infringement and Springer reserves the right to take any and all action to protect its copyright in the materials.

Copyright Notice: Disclaimer

You must include the following copyright and permission notice in connection with any reproduction of the licensed material:

"Springer book/journal title, chapter/article title, volume, year of publication, page, name(s) of author(s), (original copyright notice as given in the publication in which the material was originally published) "With permission of Springer"

In case of use of a graph or illustration, the caption of the graph or illustration must be included, as it is indicated in the original publication.

Warranties: None

Springer makes no representations or warranties with respect to the licensed material and adopts on its own behalf the limitations and disclaimers established by CCC on its behalf in its Billing and Payment terms and conditions for this licensing transaction.

Indemnity

You hereby indemnify and agree to hold harmless Springer and CCC, and their respective officers, directors, employees and agents, from and against any and all claims arising out of your use of the licensed material other than as specifically authorized pursuant to this license.

No Transfer of License

This license is personal to you and may not be sublicensed, assigned, or transferred by you without Springer's written permission.

No Amendment Except in Writing

This license may not be amended except in a writing signed by both parties (or, in the case of Springer, by CCC on Springer's behalf).

Objection to Contrary Terms

Springer hereby objects to any terms contained in any purchase order, acknowledgment, check endorsement or other writing prepared by you, which terms are inconsistent with these terms and conditions or CCC's Billing and Payment terms and conditions. These terms and conditions, together with CCC's Billing and Payment terms and conditions (which are incorporated herein), comprise the entire agreement between you and Springer (and CCC) concerning this licensing transaction. In the event of any conflict between your obligations established by these terms and conditions and those established by CCC's Billing and Payment terms and conditions, these terms and conditions shall control.

Jurisdiction
All disputes that may arise in connection with this present License, or the breach thereof, shall be settled exclusively by arbitration, to be held in the Federal Republic of Germany, in accordance with German law.

Other conditions:
V 12AUG2015

Questions? customercare@copyright.com or +1-855-239-3415 (toll free in the US) or +1-978-646-2777.

B.4 ChipWhisperer: An Open-Source Platform for Hardware Embedded Security Research [105]

SPRINGER LICENSE
TERMS AND CONDITIONS

Mar 24, 2017

This Agreement between Colin O'Flynn ("You") and Springer ("Springer") consists of your license details and the terms and conditions provided by Springer and Copyright Clearance Center.

License Number	4075590247438
License date	Mar 24, 2017
Licensed Content Publisher	Springer
Licensed Content Publication	Springer eBook
Licensed Content Title	ChipWhisperer: An Open-Source Platform for Hardware Embedded Security Research
Licensed Content Author	Colin O'Flynn
Licensed Content Date	Jan 1, 2014
Type of Use	Thesis/Dissertation
Portion	Full text
Number of copies	1
Author of this Springer article	Yes and you are the sole author of the new work
Order reference number	
Title of your thesis / dissertation	A Framework for Embedded Hardware Security Analysis
Expected completion date	Apr 2017
Estimated size(pages)	195
Requestor Location	Colin O'Flynn 1083 Queen St., Suite 196 Halifax, NS B3H0B2 Canada Attn: Colin O'Flynn
Billing Type	Invoice
Billing Address	Colin O'Flynn 1083 Queen St., Suite 196 Halifax, NS B3H0B2 Canada Attn: Colin O'Flynn
Total	0.00 USD
Terms and Conditions	

Introduction
The publisher for this copyrighted material is Springer. By clicking "accept" in connection with completing this licensing transaction, you agree that the following terms and conditions apply to this transaction (along with the Billing and Payment terms and conditions

established by Copyright Clearance Center, Inc. ("CCC"), at the time that you opened your Rightslink account and that are available at any time at http://myaccount.copyright.com).

Limited License

With reference to your request to reuse material on which Springer controls the copyright, permission is granted for the use indicated in your enquiry under the following conditions:

- Licenses are for one-time use only with a maximum distribution equal to the number stated in your request.
- Springer material represents original material which does not carry references to other sources. If the material in question appears with a credit to another source, this permission is not valid and authorization has to be obtained from the original copyright holder.
- This permission
• is non-exclusive
• is only valid if no personal rights, trademarks, or competitive products are infringed.
• explicitly excludes the right for derivatives.
- Springer does not supply original artwork or content.
- According to the format which you have selected, the following conditions apply accordingly:
• **Print and Electronic:** This License include use in electronic form provided it is password protected, on intranet, or CD-Rom/DVD or E-book/E-journal. It may not be republished in electronic open access.
• **Print:** This License excludes use in electronic form.
• **Electronic:** This License only pertains to use in electronic form provided it is password protected, on intranet, or CD-Rom/DVD or E-book/E-journal. It may not be republished in electronic open access.

For any electronic use not mentioned, please contact Springer at permissions.springer@spi-global.com.

- Although Springer controls the copyright to the material and is entitled to negotiate on rights, this license is only valid subject to courtesy information to the author (address is given in the article/chapter).
- If you are an STM Signatory or your work will be published by an STM Signatory and you are requesting to reuse figures/tables/illustrations or single text extracts, permission is granted according to STM Permissions Guidelines: http://www.stm-assoc.org/permissions-guidelines/

For any electronic use not mentioned in the Guidelines, please contact Springer at permissions.springer@spi-global.com. If you request to reuse more content than stipulated in the STM Permissions Guidelines, you will be charged a permission fee for the excess content.

Permission is valid upon payment of the fee as indicated in the licensing process. If permission is granted free of charge on this occasion, that does not prejudice any rights we might have to charge for reproduction of our copyrighted material in the future.

-If your request is for reuse in a Thesis, permission is granted free of charge under the following conditions:

This license is valid for one-time use only for the purpose of defending your thesis and with a maximum of 100 extra copies in paper. If the thesis is going to be published, permission needs to be reobtained.

- includes use in an electronic form, provided it is an author-created version of the thesis on his/her own website and his/her university's repository, including UMI (according to the definition on the Sherpa website: http://www.sherpa.ac.uk/romeo/);
- is subject to courtesy information to the co-author or corresponding author.

Geographic Rights: Scope

Licenses may be exercised anywhere in the world.

Altering/Modifying Material: Not Permitted

Figures, tables, and illustrations may be altered minimally to serve your work. You may not

alter or modify text in any manner. Abbreviations, additions, deletions and/or any other alterations shall be made only with prior written authorization of the author(s).

Reservation of Rights

Springer reserves all rights not specifically granted in the combination of (i) the license details provided by you and accepted in the course of this licensing transaction and (ii) these terms and conditions and (iii) CCC's Billing and Payment terms and conditions.

License Contingent on Payment

While you may exercise the rights licensed immediately upon issuance of the license at the end of the licensing process for the transaction, provided that you have disclosed complete and accurate details of your proposed use, no license is finally effective unless and until full payment is received from you (either by Springer or by CCC) as provided in CCC's Billing and Payment terms and conditions. If full payment is not received by the date due, then any license preliminarily granted shall be deemed automatically revoked and shall be void as if never granted. Further, in the event that you breach any of these terms and conditions or any of CCC's Billing and Payment terms and conditions, the license is automatically revoked and shall be void as if never granted. Use of materials as described in a revoked license, as well as any use of the materials beyond the scope of an unrevoked license, may constitute copyright infringement and Springer reserves the right to take any and all action to protect its copyright in the materials.

Copyright Notice: Disclaimer

You must include the following copyright and permission notice in connection with any reproduction of the licensed material:

"Springer book/journal title, chapter/article title, volume, year of publication, page, name(s) of author(s), (original copyright notice as given in the publication in which the material was originally published) "With permission of Springer"

In case of use of a graph or illustration, the caption of the graph or illustration must be included, as it is indicated in the original publication.

Warranties: None

Springer makes no representations or warranties with respect to the licensed material and adopts on its own behalf the limitations and disclaimers established by CCC on its behalf in its Billing and Payment terms and conditions for this licensing transaction.

Indemnity

You hereby indemnify and agree to hold harmless Springer and CCC, and their respective officers, directors, employees and agents, from and against any and all claims arising out of your use of the licensed material other than as specifically authorized pursuant to this license.

No Transfer of License

This license is personal to you and may not be sublicensed, assigned, or transferred by you without Springer's written permission.

No Amendment Except in Writing

This license may not be amended except in a writing signed by both parties (or, in the case of Springer, by CCC on Springer's behalf).

Objection to Contrary Terms

Springer hereby objects to any terms contained in any purchase order, acknowledgment, check endorsement or other writing prepared by you, which terms are inconsistent with these terms and conditions or CCC's Billing and Payment terms and conditions. These terms and conditions, together with CCC's Billing and Payment terms and conditions (which are incorporated herein), comprise the entire agreement between you and Springer (and CCC) concerning this licensing transaction. In the event of any conflict between your obligations established by these terms and conditions and those established by CCC's Billing and Payment terms and conditions, these terms and conditions shall control.

Jurisdiction

All disputes that may arise in connection with this present License, or the breach thereof,

shall be settled exclusively by arbitration, to be held in the Federal Republic of Germany, in accordance with German law.

Other conditions:

V 12AUG2015

B.5 A case study of Side-Channel Analysis using Decoupling Capacitor Power Measurement with the OpenADC [101]

SPRINGER LICENSE
TERMS AND CONDITIONS

Mar 24, 2017

This Agreement between Colin O'Flynn ("You") and Springer ("Springer") consists of your license details and the terms and conditions provided by Springer and Copyright Clearance Center.

License Number	4075590628366
License date	Mar 24, 2017
Licensed Content Publisher	Springer
Licensed Content Publication	Springer eBook
Licensed Content Title	A Case Study of Side-Channel Analysis Using Decoupling Capacitor Power Measurement with the OpenADC
Licensed Content Author	Colin O'Flynn
Licensed Content Date	Jan 1, 2013
Type of Use	Thesis/Dissertation
Portion	Full text
Number of copies	1
Author of this Springer article	Yes and you are the sole author of the new work
Order reference number	
Title of your thesis / dissertation	A Framework for Embedded Hardware Security Analysis
Expected completion date	Apr 2017
Estimated size(pages)	195
Requestor Location	Colin O'Flynn 1083 Queen St., Suite 196 Halifax, NS B3H0B2 Canada Attn: Colin O'Flynn
Billing Type	Invoice
Billing Address	Colin O'Flynn 1083 Queen St., Suite 196 Halifax, NS B3H0B2 Canada Attn: Colin O'Flynn
Total	0.00 USD
Terms and Conditions	

Introduction
The publisher for this copyrighted material is Springer. By clicking "accept" in connection with completing this licensing transaction, you agree that the following terms and conditions apply to this transaction (along with the Billing and Payment terms and conditions

established by Copyright Clearance Center, Inc. ("CCC"), at the time that you opened your Rightslink account and that are available at any time at http://myaccount.copyright.com).

Limited License

With reference to your request to reuse material on which Springer controls the copyright, permission is granted for the use indicated in your enquiry under the following conditions:

- Licenses are for one-time use only with a maximum distribution equal to the number stated in your request.
- Springer material represents original material which does not carry references to other sources. If the material in question appears with a credit to another source, this permission is not valid and authorization has to be obtained from the original copyright holder.
- This permission
• is non-exclusive
• is only valid if no personal rights, trademarks, or competitive products are infringed.
• explicitly excludes the right for derivatives.
- Springer does not supply original artwork or content.
- According to the format which you have selected, the following conditions apply accordingly:
• **Print and Electronic:** This License include use in electronic form provided it is password protected, on intranet, or CD-Rom/DVD or E-book/E-journal. It may not be republished in electronic open access.
• **Print:** This License excludes use in electronic form.
• **Electronic:** This License only pertains to use in electronic form provided it is password protected, on intranet, or CD-Rom/DVD or E-book/E-journal. It may not be republished in electronic open access.

For any electronic use not mentioned, please contact Springer at permissions.springer@spi-global.com.

- Although Springer controls the copyright to the material and is entitled to negotiate on rights, this license is only valid subject to courtesy information to the author (address is given in the article/chapter).
- If you are an STM Signatory or your work will be published by an STM Signatory and you are requesting to reuse figures/tables/illustrations or single text extracts, permission is granted according to STM Permissions Guidelines: http://www.stm-assoc.org/permissions-guidelines/

For any electronic use not mentioned in the Guidelines, please contact Springer at permissions.springer@spi-global.com. If you request to reuse more content than stipulated in the STM Permissions Guidelines, you will be charged a permission fee for the excess content.

Permission is valid upon payment of the fee as indicated in the licensing process. If permission is granted free of charge on this occasion, that does not prejudice any rights we might have to charge for reproduction of our copyrighted material in the future.

-If your request is for reuse in a Thesis, permission is granted free of charge under the following conditions:

This license is valid for one-time use only for the purpose of defending your thesis and with a maximum of 100 extra copies in paper. If the thesis is going to be published, permission needs to be reobtained.

- includes use in an electronic form, provided it is an author-created version of the thesis on his/her own website and his/her university's repository, including UMI (according to the definition on the Sherpa website: http://www.sherpa.ac.uk/romeo/);
- is subject to courtesy information to the co-author or corresponding author.

Geographic Rights: Scope

Licenses may be exercised anywhere in the world.

Altering/Modifying Material: Not Permitted

Figures, tables, and illustrations may be altered minimally to serve your work. You may not

alter or modify text in any manner. Abbreviations, additions, deletions and/or any other alterations shall be made only with prior written authorization of the author(s).

Reservation of Rights

Springer reserves all rights not specifically granted in the combination of (i) the license details provided by you and accepted in the course of this licensing transaction and (ii) these terms and conditions and (iii) CCC's Billing and Payment terms and conditions.

License Contingent on Payment

While you may exercise the rights licensed immediately upon issuance of the license at the end of the licensing process for the transaction, provided that you have disclosed complete and accurate details of your proposed use, no license is finally effective unless and until full payment is received from you (either by Springer or by CCC) as provided in CCC's Billing and Payment terms and conditions. If full payment is not received by the date due, then any license preliminarily granted shall be deemed automatically revoked and shall be void as if never granted. Further, in the event that you breach any of these terms and conditions or any of CCC's Billing and Payment terms and conditions, the license is automatically revoked and shall be void as if never granted. Use of materials as described in a revoked license, as well as any use of the materials beyond the scope of an unrevoked license, may constitute copyright infringement and Springer reserves the right to take any and all action to protect its copyright in the materials.

Copyright Notice: Disclaimer

You must include the following copyright and permission notice in connection with any reproduction of the licensed material:

"Springer book/journal title, chapter/article title, volume, year of publication, page, name(s) of author(s), (original copyright notice as given in the publication in which the material was originally published) "With permission of Springer"

In case of use of a graph or illustration, the caption of the graph or illustration must be included, as it is indicated in the original publication.

Warranties: None

Springer makes no representations or warranties with respect to the licensed material and adopts on its own behalf the limitations and disclaimers established by CCC on its behalf in its Billing and Payment terms and conditions for this licensing transaction.

Indemnity

You hereby indemnify and agree to hold harmless Springer and CCC, and their respective officers, directors, employees and agents, from and against any and all claims arising out of your use of the licensed material other than as specifically authorized pursuant to this license.

No Transfer of License

This license is personal to you and may not be sublicensed, assigned, or transferred by you without Springer's written permission.

No Amendment Except in Writing

This license may not be amended except in a writing signed by both parties (or, in the case of Springer, by CCC on Springer's behalf).

Objection to Contrary Terms

Springer hereby objects to any terms contained in any purchase order, acknowledgment, check endorsement or other writing prepared by you, which terms are inconsistent with these terms and conditions or CCC's Billing and Payment terms and conditions. These terms and conditions, together with CCC's Billing and Payment terms and conditions (which are incorporated herein), comprise the entire agreement between you and Springer (and CCC) concerning this licensing transaction. In the event of any conflict between your obligations established by these terms and conditions and those established by CCC's Billing and Payment terms and conditions, these terms and conditions shall control.

Jurisdiction

All disputes that may arise in connection with this present License, or the breach thereof,

shall be settled exclusively by arbitration, to be held in the Federal Republic of Germany, in accordance with German law.

Other conditions:

V 12AUG2015