Side Channel Analysis of an Embedded/Hardware Crypto Device

Dallin Marshall, Sam Mitchell, and Nathanael Weidler
Deptartment of Electrical and Computer Engineering
Utah Stat University
Logan, Utah 84322
e-mail: geekbott@gmail.com, samuel.alan.mitchell@gmail.com, NWeidler@gmail.com

***Abstract*—This paper describes the design and implementation of a physically unclonable function (PUF).**

***Index Terms*—Physically unclonable Function, Device Authentication.**

I. Introduction

A physically unclonable function (PUF) is one method of testing user authentication. The server sends a challenge to a device, and the device responds using the output to the PUF; this is called a challenge response pair. If the PUF is truly unclonable, this authentication method is effectively secure.

The analysis of PUFs in authentication relies on 2 characteristics of the PUF: the variation between devices, μ_{intra}, and the reliability of the device to reproduce the same bitstream given a challenge, μ_{inter}.

A. Related works

There are various types of PUFs that can be implemented on an FPGA [2].

[2] Types of memory-based PUFs.

- SRAM
- Latch (Butterfly) [1]
- Flip-flop [3], [4]
- Buskeeper [6]

There are 3 main purposes of PUFs,

B. Structure of paper

The organization is as follows: in Section II, the development of the PUF is presented. In Section III the experimental test setup for the capturing of the challenge response pairs is described. Section IV contains the analysis of the data. Conclusions are detailed in Section VI.

II. Design

The butterfly PUF consists of wiring 2 NAND gates together in positive feedback mode, as shown in Figure 1. Upon excitation, the gates are put into a race condition until the voltage level converges (less than 1 second). The butterfly should result in a consistent output for each device it's implemented on, while the output of the devices have low correlation. When implemented on a Spartan IV, the majority of the butterfly PUFs gave consistent but unique output, but the logic of 12% never converges to one voltage.

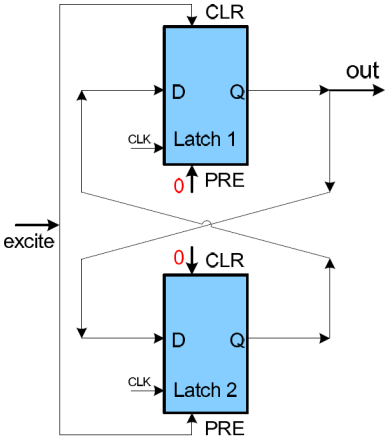

Figure 1. Butterfly PUF: Cross-coupled latches.

This paper utilizes a modified butterfly PUF that accepts an input. The excite signal is used as a switching signal through negation, as shown in Figure 2. This allows the PUF to accept an input, which results in 2 potential random outputs.

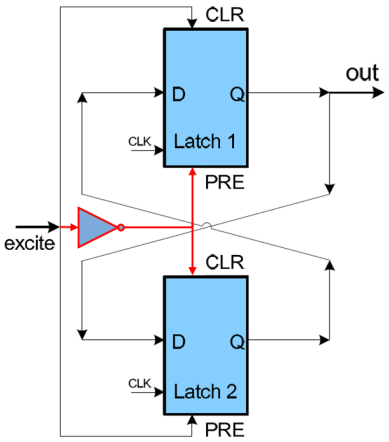

Figure 2. Modified butterfly PUF: Cross-coupled latches are modifiable.

A. Butterfly in authentication

An authentication device accepts a challenge sequence and sends back a device-specific response.

B. Butterfly in TRNG

As a part of this work an implementation of DES was written for the Tiva-C microprocessor. "The DES Algorith Illustrated" was a valuable resource during this process. All of the explicit details of DES will be left to other work, however, a brief overview will be presented in order for the reader to grasp the complexity of the algorithm.

DES is a 16 round encryption algorithm. It works on 64-bit blocks of plaintext and outputs encrypted data of the same size. The output of the algorithm is the cipher text. The first step in DES is to divide the plaintext block in half, the left 32-bits and the right 32-bits. The heart of DES is the 56-bit master key which is used to create 16 distinct 48-bit round keys, one for each round of the algorithm. (The intent of this attack is to recover the key from the 16th round, from which all other keys can be easily obtained.) To create each round key the 56-bit key is permuted and divided into two halves. Then for each round each of those halves is shifted, recombined and permuted yielding the 48-bit round keys.

One half of the plaintext (32-bits) is expanded and permuted to create a 48-bit word which is xored with the round key. The result is divided into eight 6-bit parts which go through 8 distinct s-boxes. An s-box is method to substitute a 6-bit input with a 4-bit output. These 4-bit outputs are put back together and permuted to create a 32-bit word which is xored with the other half of the plaintext. This process is repeated 16 times until finally the two halves are put back together and permuted one last time to create the cipher text. An illustration of one round is shown in figure 3.

Figure 3. One round of DES.

C. DES Modifications

In order to facilitate the process by which traces are obtained, there was an addition to the traditional DES algorithm. This slight modification in no way compromises the integrity of the algorithm or its outputs. The change was to hold a general purpose input/output (GPIO) high during the algorithm except during a single instruction in which the output of the Feistel function is xored with the other half of the plaintext in round 16 and the result is written back to a register. This allowed an oscilloscope to be triggered on the falling edge of the GPIO and frame this critical step which will be discussed later. There were also two no-operations (nops) added before this register write and one after to allow the GPIO to settle and facilitate a cleaner capture. The output of the assembly dump with comments added can be seen in figure 4.

Figure 4. The assembly showing the GPIO writes surrounding the xor and register write under attack.

III. Experimental Setup

This portion of the experiment was the most crucial to its success. The goal is to obtain information about the power usage of the microprocessor during a critical step of the DES algorithm. During a register write operation the power used will be different depending on the value of the word written. It is believed that by analyzing this information, the secret key to the algorithm can be obtained.

The oscilloscope used was a Techtronix DP0-2024 the sample rate was 1 giga-sample per second and each trace contained about 270 samples. The Tiva-C used a 17 MHz clock so there 38.5 samples per clock.

The setup used involved the Tiva-C microprocessor powered by a bench top power supply through a breadboard. The ground connection between the microprocessor and the power supply had a 1-ohm resistor added in series. An Oscilloscope probe was placed on the resistor so that the voltage could be measured. Then another oscilloscope probe was placed on the GPIO pin discussed in the previous section to act as the trigger. The voltage data would be taken twenty times for each plain text value used in the DES algorithm and then averaged in an attempt to reduce noise. 91,840 captures with 4,592 unique plain text values would be obtained. Each trace takes about two seconds to capture so the time to obtain all traces is 183,680 seconds or about 51 hours, or 2.13 days. The voltage data acquired would be analyzed later to determine the secret key.

The test setup needed to be automated to facilitate the capturing of the traces. In order for this to be done a matlab file was written to interface with the oscilloscope. As trivial as this may sound it was not easy. It took many hours of learning the oscilloscope and the commands to control it through matlab. It also took time and patience to fine tune the DES implementation to create enough delay between each run of the algorithm to allow for the oscilloscope to transfer the data capture and re-arm itself.

IV. Data Analysis

A. Differential Power Analysis

In order to analyze the data, differential power analysis (DPA) was employed. Round 16 was chosen for this because given the cipher text, the input to the Feistel function and the output from the second xor (seen in the bottom right of figure 3) can be determined. Then the round key is guessed six bits at a time. This greatly reduces the complexity of the key as there are only 64 possible combinations for a guess of six bits of the key. Using knowledge of the Feistel function and the six bit key guess, the output of the Feistel function can be guessed at four bits at a time. Remember, the s-boxes reduce the data from 48 bits to 32 bits. This output is xored with the D value, and we already know the output of this xor. So guessing one input and knowing the output determines the other input to the xor gate.

For each key guess four bits of the D word are deduced. These 4-bit groups are divided into 3 sets where set 0 contains all guesses for which the four bits guessed are "0000," set 1

contains all guess where the four bits are "1111," and set 2 contains all other guessed values. The average power for each sample point for sets 0 and 1 is calculated. The two average powers are subtracted from each other. The correct key guess will show large spikes where the calculation was actually made on the trace. This process is repeated eight times, once for each s-box until the correct key is determined.

B. Correlation Power Analysis

fill in here...

C. Actual Key Values Analysed

This experiment is unique from an actual side channel analysis attack because we knew what the secret key was. This meant that we could run the correct values through DPA and CPA as was done previously with guesses. The results will be discussed in the next section.

V. Results

The results were not as promising as we had hoped they would be. Simply because we knew what the secret key was we could potentially recover parts of the secret key.

VI. Conclusion

As can be seen, side channel analysis can be an effective means to recover data from embedded systems. DPA has previously been shown to work. It seems the determining factor of whether or not your side channel analysis will be a success is the acquisition of the.

References

[1] S. Kumar, J. Guajardo, R. Maes, G.-J. Schrijen, and P. Tuyls, "Extended abstract: The butterfly puf protecting ip on every fpga," in *Hardware-Oriented Security and Trust, 2008. HOST 2008. IEEE International Workshop on*, June 2008, pp. 67–70.

[2] A. Van Herrewege, "Lightweight puf-based key and random number generation," 2015.

[3] V. van der Leest, G.-J. Schrijen, H. Handschuh, and P. Tuyls, "Hardware intrinsic security from d flip-flops," in *Proceedings of the fifth ACM workshop on Scalable trusted computing*. ACM, 2010, pp. 53–62.

[4] R. Maes, P. Tuyls, and I. Verbauwhede, "Intrinsic pufs from flip-flops on reconfigurable devices," in *3rd Benelux workshop on information and system security (WISSec 2008)*, vol. 17, 2008.

[5] M. Platonov, J. Hlavác, and R. Lórencz, "Using power-up sram state of atmel atmega1284p microcontrollers as physical unclonable function for key generation and chip identification," *Information Security Journal: A Global Perspective*, vol. 22, no. 5-6, pp. 244–250, 2013.

[6] P. Simons, E. van der Sluis, and V. van der Leest, "Buskeeper pufs, a promising alternative to d flip-flop pufs," in *Hardware-Oriented Security and Trust (HOST), 2012 IEEE International Symposium on*, June 2012, pp. 7–12.