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

Some of the main usages of PUFs are to provide reliable device identification, device authentication, key storage, and true random number generation [1].

Not all PUFs can be implemented on every type of device (FPGA, ASIC, microcontroller). There are various types of PUFs that can be implemented on an FPGA [2], some of which are SRAM, Butterfly [3], flip-flop [4], [5], Buskeeper [6].

The Butterfly PUF was analyzed in [7] with results inconsistent with [3]. The authors determined that this was due to the difficulty to define the routing distance between gates.

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

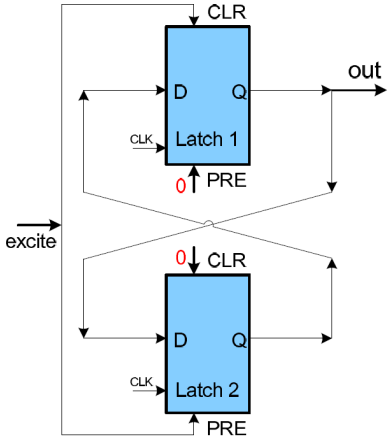

Figure 1. Butterfly PUF: This works because of cross-coupled latches.

of the butterfly PUFs gave consistent but unique output, but the logic of 12% never converges to one voltage.

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. Testing showed that some PUFs would consistently toggle, some would consistently remain 1 value, and some weren't consistent, which is consistent with the 12% of the standard butterfly PUF that wouldn't converge. This PUF is very compact, requiring only 0.03% of the FPGA space.

A. Butterfly in authentication

An authentication device accepts a challenge sequence and sends back a device-specific response. We designed a modified-butterfly PUF authentication device that accepts a 64-bit key input and responds with a 64-bit key.

The device is a 64 by 64 grid of modified-butterfly PUFs. Each column uses an XOR gate to accept 1 bit of the 64-bit key; column 1 accepts bit 1, the output of column 1 is XORed with bit 2 and fed to column 2. The row order corresponds with the response bits; row 1 produces bit 1.

This design is valid because each bit of the key can alter the response up to 50%, which means that the corresponding response bit can only be predicted with 50% accuracy. Each

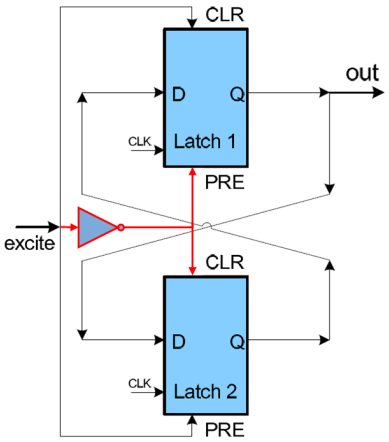

Figure 2. Modified-butterfly PUF: The cross-coupled latches are altered by the input.

row has a unique string of PUFs, so each response bit will be random to the other bits.

A simple verification was performed on this system by implementing 1 row of our modified-butterfly authenticator PUF. Unique keys successfully produced unique responses, but the responses were consistent only 83% of the time. Without error correction coding, the system will never correctly produce the correct response (0.0064% likelihood). The modified-butterfly PUF was discarded for authentication applications.

B. Butterfly in TRNG

TRNG requires random processes to extract numbers from. Both versions of the butterfly PUF produce a race condition before converging to one value. This race condition will quickly settle unless perturbed.

Two modified-butterfly PUFs are wired together to induce race conditions in each PUF. This is done by connecting the output of a modified-butterfly PUF to the input of another modified-butterfly PUF, and vice versa. The output of one of the PUFs is sampled periodically (at the communication rate) to produce a stream of random bits.

III. Experimental Setup

IV. Data Analysis

V. Results

VI. Conclusion

References

[1] M. Tehranipoor and C. Wang, *Introduction to hardware security and trust*. Springer Science & Business Media, 2011.

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

[3] 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.

[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] 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.

[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.

[7] S. Morozov, A. Maiti, and P. Schaumont, "An analysis of delay based puf implementations on fpga," in *Reconfigurable Computing: Architectures, Tools and Applications*. Springer, 2010, pp. 382–387.