Physical Unclonable Function with Tristate Buffers

Erdinc Ozturk
Worcester Polytechnic Institute
Worcester MA, 01609, USA
erdinc@wpi.edu

Ghaith Hammouri
Worcester Polytechnic Institute
Worcester MA, 01609, USA
hammouri@wpi.edu

Berk Sunar
Worcester Polytechnic Institute
Worcester MA, 01609, USA
sunar@wpi.edu

***Abstract*— The lack of robust tamper-proofing techniques in security applications has provided attackers the ability to virtually circumvent mathematically strong cryptographic primitives by directly attacking the hardware. Consequently, physical tamper-proofing has emerged as an essential element in secure system design. To this end, physical unclonable functions (PUF) have been proposed to build tamper proof hardware and thereby create secure data storages. A delay based PUF scheme was proposed which uses the intrinsic process variations to randomize switches and thereby implement a pseudorandom function family. When tampered with, the device experiences a change in its internal physical parameters. This will alter the pseudorandom function family. Therefore, rendering an attack unsuccessful. In this paper, we propose a delay based PUF circuit that is constructed from tristate buffers. The proposed PUF circuit consumes less power and requires less area. Furthermore, we develop a linear delay model which turns out to be identical to that of switch based PUFs. Hence, the nonlinearization techniques proposed to secure switch based PUFs can directly be applied to secure tristate PUFs as well.**

I. Introduction

Recently we have witnessed an explosive growth in the type and strength of attacks on secured devices. These attacks circumvent the data channel and instead exploit so called side–channels. These attacks are classified into two groups: passive and active attacks. Passive attacks solely observe side–channels (e.g. computation time, power consumption, electromagnetic emanation, temperature attacks etc.) to deduce internal secrets from leaked side–channel profiles. A stronger group of attacks may be classified as active attacks. To induce an active attack, the attacker may also inject faults during the computation and subsequently mount a passive attack [1]. Active attacks are more powerful and are more difficult to prevent. Building a tamper–proof hardware is crucial in securing devices from both passive and active side–channel attacks. The use of Physically Unclonable Functions (PUFs) has been proposed to build a tamper–proof hardware [2]. A PUF is a physical pseudo–random function which may be implemented by exploiting the small variances of the wire and gate delays that are unique for each integrated circuit (IC), even if they are logically identical. These variances depend on highly unpredictable factors, which are mainly caused by the inter–chip manufacturing variations. Hence, given the same input, the PUF circuit will return a different output on different chipsets. Additionally, if the PUFs are manufactured with high precision, any external physical influence will change the PUF function and render the device non–functional. For instance, trying to tap the PUF circuit will change the loading capacitance. Even de–layering the IC will effect the wire delays thus changing the output of the PUF circuit. Briefly, it can be assumed that a properly manufactured PUF device will be tamper–resilient even for fairly sophisticated attack schemes.

II. Related Work

A switch–based PUF with an n–bit challenge input C and a 1–bit response output R consists of n stages of switching circuits. Each switch has two input and two output bits. If the control bit of the switch (respective challenge bit) is logical 0, the input is mirrored to the output. If it is 1, the two paths are switched. This is a way of switching the delay paths. In the switch–based delay circuit, the

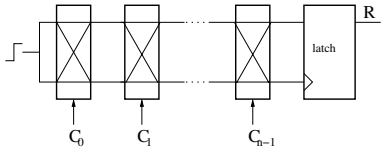


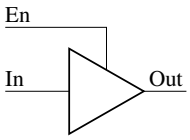
Fig. 1. A switch based PUF circuit

978-1-4244-1684-4/08/$25.00 ©2008 IEEE

output of each switch is connected to the input of the consecutive switch. In the end, the two output bits of the last switch are connected to a flip–flop, which is called the *arbiter*. If the path that is connected to the data input of the arbiter has a smaller delay, then the output will be 1. Otherwise, the output will be 0. This interpretation is accurate only if we assume that the setup time of the arbiter is negligible compared to the delay differences.

III. Proposed Tristate PUF Circuit

The original PUF circuit utilizes multiplexers (MUXs) to implement the switches. Each delay unit includes two MUX circuits. Instead of having two interleaved delay paths, we propose a circuit with two separate delay paths. We construct our PUF circuit utilizing the delay variances of tristate buffers.



En	In	Out
0	X	Hi–Z
1	0	0
1	1	1

Fig. 2. A generic tristate buffer

A. Tristate Buffers

A generic tristate buffer as shown in Figure 2 has three states: logic 1, logic 0 and high impedance (referred to as Hi–Z). If the gate is enabled, the input will be reflected at the output. If not enabled, the output will become Hi–Z. The truth table of a tristate buffer is shown in Figure 2. The outputs of two tristate buffers can be connected together to behave as a multiplexer gate. However, when both of the tristate buffers are enabled, the circuit is under risk of being damaged. If the inputs of the two tristate buffers are complement values of each other, and the enable inputs are both high, then the circuit will start heating up and eventually will be damaged.

B. Delay Unit

A PUF circuit is based on the delay characteristics of the logic design. We build our tristate buffer based on the delay unit shown in Figure 3. Each delay unit is constructed by respectively connecting the input and output ports of the two tristate buffers. The enable ports of these tristate buffers are connected to each other, with one of the buffers having an inverted enable input. This

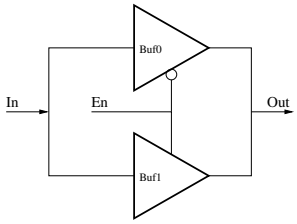


Fig. 3. Delay unit built with tristate buffers

assures that only one of the tristate buffer gates will be enabled for a particular enable input value. When a pulse is applied at the input of this delay unit, it will pass through either Buf0 or Buf1. The enable input will determine which tristate buffer passes the pulse. This will change the amount of the delay that this unit contributes according to the value of the enable signal.

C. PUF circuit

The delay units are cascaded to construct a serial delay path. For our PUF circuit, we need two separate delay paths. These two delay paths are connected as shown in Figure 4. The inputs of these delay paths are connected together and the outputs are fed to an arbiter. The arbiter will capture the faster path by producing logic 1 or 0 at the output. Assume the arbiter is built with a positive edge–triggered flip–flop. The upper path is connected to the data input of the flip–flop and the lower path is connected to its clock input. If the lower path is faster, the rising edge of the clock signal will arrive before the rising edge of the data signal, therefore producing a logic 0 at the arbiter output. Alternatively, if the upper path is faster we observe the opposite effect, i.e. the output of the arbiter becomes a logic 1.

IV. Security Analysis

In this section we show that a tristate PUF is equivalent to a switch based PUF in terms of security. In particular we show that both implementations can be modeled using a similar linear model. In the tristate implementation of a PUF there are two groups of consecutive delay units; the upper and the lower. If we consider the upper path we notice that in each delay unit there are two possible paths. Let us call the delay in each of these two paths a_i and b_i, where i is the stage index. Let $H_i = \frac{a_i+b_i}{2}$ and $y_i = \frac{a_i-b_i}{2}$, which means that $a_i = H_i + y_i$ and $b_i = H_i - y_i$. Depending on whether the challenge bit is 0 or 1, the signal going through the upper units will be delayed by

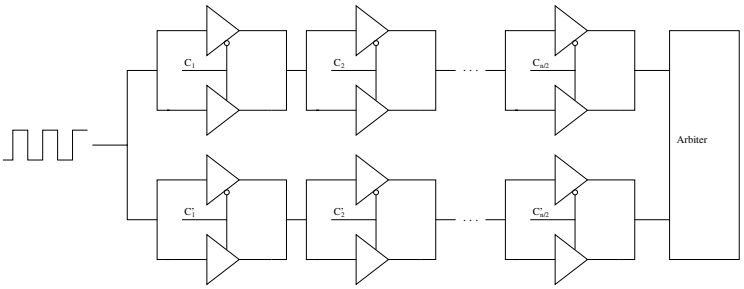


Fig. 4. PUF architecture built with tristate buffers

$H_i + (-1)^{c_i} y_i$ where c_i is the i^{th} challenge to the upper delay units. Assuming that the signal has $n/2$ stages to propagate through, the total delay in the upper delay units becomes equal to

$$D_H = \sum_{i=1}^{n/2} H_i + (-1)^{c_i} y_i = \tau_H + \sum_{i=1}^{n/2} (-1)^{c_i} y_i \quad , \tag{1}$$

where $\tau_H = \sum_{i=1}^{n/2} H_i$. Similarly, one can derive the delay equation for the lower delay units. The two delays in each of the paths of the lower delay units can be named d_i and f_i. Similarly one can make $L_i = \frac{d_i + f_i}{2}$ and $u_i = \frac{d_i - f_i}{2}$. Now if we let c_i' be the challenge bit to the ith stage of the lower units, the delay in each unit becomes $L_i + (-1)^{c_i'} u_i$. Therefore, the total delay in the lower path becomes:

$$D_L = \sum_{i=1}^{n/2} L_i + (-1)^{c_i'} u_i = \tau_L + \sum_{i=1}^{n/2} (-1)^{c_i'} u_i \quad . \tag{2}$$

The two signals traveling through the upper and lower delay units will interact through the arbiter. The condition on the output bit becomes:

$$D_H < D_L + T_S \rightarrow R = 1$$
$$D_H > D_L + T_S \rightarrow R = 0 \quad ,$$

where T_S is the setup time for the arbiter latch. One can make the two relations more concise by relating the response bit, $(-1)^R (D_L - D_H + T_s) < 0$. If we incorporate the original equations for D_L and D_H, we get the following response equation:

$$(-1)^R (\sum_{i=1}^{n/2} (-1)^{c_i'} u_i - (-1)^{c_i} y_i + T) < 0 \quad , \tag{3}$$

where $T = \tau_L - \tau_H + T_S$. Note that T, y_i and u_i are variables dependent on the circuit. This model is identical of that used in the switch based PUF [3], where the delay equation can be related to the response bit using the following equation:

$$(-1)^R (\sum_{i=1}^{n} (-1)^{P_i} y_i + T) < 0 \quad , \tag{4}$$

where $P_i = c_i \oplus c_{i-1} \oplus \ldots \oplus c_n$. It can easily be seen that the two equations are mathematically equivalent. Both implementations can be characterized using the same number of variables that are related to each other using identical equations. Equations 3 and 4 are both linear inequalities with $n + 1$ variables. From the point of view of the attacker, such an inequality can be produced for every observed challenge–response pair, thus yielding a system of linear inequalities. The problem of solving systems of linear inequalities is crucial in a wide variety of areas and therefore has been well developed providing a rich family of algorithms commonly referred to under linear programming. One can easily run these algorithms on a large set of challenge–response pairs in order to obtain an accurate estimation of the $n+1$ variables characterizing the switch or the tristate PUF implementations.

In order to test our mathematical model, we ran a number of tests using MATLAB. Our tests were performed for three sizes of a PUF circuit $n = 32, 64$ and 128. For each n we generated the $n + 1$ variables characterizing the PUF using a Gaussian distribution, with a mean of 0 and a standard deviation of 1. For each n we generated a number of random challenges N_s, and then calculated the response of the PUF according to the model presented in this paper. We then used the challenge–response pairs generated in order to solve for the $n+1$ variables which were randomly chosen. In order to solve these equations we used the medium scale algorithm supported by MATLAB [4], this algorithm is a variation of the known algorithm Simplex [5].

In order to test the viability of the model, we generated 500,000 random points and compared the

	Number of challenge–response pairs N_s									
n	32	64	128	256	512	1024	2048	4096	8192	16384
32	40.33	33.03	21.18	11.63	5.70	2.61	1.37	0.61	0.28	0.14
64	45.77	40.62	32.44	21.27	11.30	5.64	2.60	1.11	0.57	0.27
128	47.66	45.30	40.08	32.02	21.38	12.13	5.94	2.89	1.27	0.59

TABLE I

PERCENTAGE ERROR FOR EACH (n, N_s) PAIR

response produced by our linear model. Table I shows the percentage error we obtained. Our discussion here was mainly directed towards showing that the tristate and switch based implementations of a PUF are indeed equivalent, and can be modeled and simulated to a high accuracy in the same way. However, this ability to accurately model the PUF circuit will compromise the security of the scheme, especially with such high level of accuracy. As this observation applies to the switch based implementation as well as the tristate implementation, there have been multiple methods put forward in order to resist such modeling attacks [6]. Most of these models rely on inserting a source of non-linearity in the circuit, thus disturbing the linear model. As these techniques are not specific to the switch based implementation, they can easily be adapted to the the tristate implementation, therefore retaining both implementations at the same level of security.

V. IMPLEMENTATION RESULTS

We implemented the proposed tristate PUF and compared it against an implementation of a switch–based PUF built using multiplexers. The I/O for both circuits consisted of 64-bit challenge inputs, a single bit pulse input and a response output. Both designs were developed into Verilog modules and synthesized using the Synopsys tools Design Compiler and Power Compiler with the TSMC 0.13 μm ASIC library. The results are shown in Table II.

	Total Power(uW)		Area(Gates)
	10MHz	100MHz	
Tristate PUF	18.78	152.93	351
Mux PUF	23.14	193.67	450

TABLE II

SYNTHESIS RESULTS

The implementation results clearly display the superiority of the tristate implementation over the typical switch-based implementation in terms of design footprint and power consumption. These are attractive features for low-power devices such as RFIDs, smart cards and sensor networks. Note that the area (and the power) consumption of the tristate PUF may be further reduced by custom CMOS design.

VI. CONCLUSION

We propose a delay based PUF circuit that is constructed from tristate buffers. Our simulation results show that the proposed PUF circuit consumes less power (23% at 100 MHz, 18% at 10 MHz) and requires 23% less area compared to the MUX based version. Furthermore, we develop a linear delay model which turns out to be identical to that of the switch based PUFs. Hence, the nonlinearization techniques proposed to secure switch based PUFs can immediately be applied to secure tristate PUFs. We set as future work the IC implementation of the two delay based PUFs and verification of our simulation results.

VII. ACKNOWLEDGMENTS

This material is based upon work supported by the National Science Foundation under Grant No. ANI-0133297 (NSF CAREER Award).

REFERENCES

[1] K. J. Kulikowski, M. G. Karpovsky, and A. Taubin, "Dpa on faulty cryptographic hardware and countermeasures," in *FDTC*, 2006, pp. 211–222.

[2] B. Gassend, D. Clarke, M. van Dijk, and S. Devadas, "Delay-based Circuit Authentication and Applications," in *Proceedings of the 2003 ACM Symposium on Applied Computing*, 2003, pp. 294–301.

[3] B. Gassend, D. Lim, D. Clarke, M. van Dijk, and S. Devadas, "Identification and authentication of integrated circuits: Research articles," *Concurr. Comput. : Pract. Exper.*, vol. 16, no. 11, pp. 1077–1098, 2004.

[4] G. B. Dantzig, A. Orden, and P. Wolfe, "The generalized simplex method for minimizing a linear form under linear inequality restraints," vol. 5, no. 2, pp. 183–195, 1955.

[5] E. D. Andersen and K. D. Andersen, "Presolving in linear programming," *Mathematical Programming*, vol. 71, no. 2, pp. 221–245, 1995.

[6] J. W. Lee, D. Lim, B. Gassend, G. E. Suh, M. van Dijk, and S. Devadas, "A technique to build a secret key in integrated circuits for identification and authentication application," in *Proceedings of the Symposium on VLSI Circuits*, 2004, pp. 176–159.