Silicon Physical Random Functions*

Blaise Gassend, Dwaine Clarke, Marten van Dijk† and Srinivas Devadas
Massachusetts Institute of Technology
Laboratory for Computer Science
Cambridge, MA 02139, USA

{gassend,declarke,marten,devadas}@mit.edu

ABSTRACT

We describe the notion of a Physical Random Function (PUF). We argue that a complex integrated circuit can be viewed as a silicon PUF and describe a technique to identify and authenticate individual integrated circuits (ICs).

We describe several possible circuit realizations of different PUFs. These circuits have been implemented in commodity Field Programmable Gate Arrays (FPGAs). We present experiments which indicate that reliable authentication of individual FPGAs can be performed even in the presence of significant environmental variations.

We describe how secure smart cards can be built, and also briefly describe how PUFs can be applied to licensing and certification applications.

Categories and Subject Descriptors

C.3 [**Special-Purpose and Application-Based Systems**]: Smartcards

General Terms

Measurement, Experimentation, Security

Keywords

Physical security, unclonability

1. INTRODUCTION

We describe the notion of Physical Random Functions (PUFs) and argue that PUFs can be implemented using conventional integrated circuit (IC) design techniques. This leads us to a method of identifying and authenticating individual ICs and a means of building secure smartcards. A host of other applications are also possible.

Many methods are already available to identify and authenticate ICs. One can embed a unique identifier in an IC to give it a unique identity. This approach can identify the IC, but cannot authenticate it. To enable authentication, one needs to embed a secret key onto the IC. Of course, for the system to work, this key needs to remain secret, which means that the packaged IC has to be made resistant to attacks that attempt to discover the key. Numerous attacks are described in the literature. These attacks may be invasive, e.g., removal of the package and layers of the IC, or non-invasive, e.g., differential power analysis that attempts to determine the key by stimulating the IC and observing the power and ground rails. Making an IC tamper-resistant to all forms of attacks is a challenging problem and is receiving some attention [1]. IBM's PCI Cryptographic Coprocessor encapsulates a 486-class processing subsystem within a tamper-sensing and tamper-responding environment where one can run security-sensitive processes [12]. However, providing high-grade tamper resistance, which makes it impossible for an attacker to access or modify the secrets held inside a device, is expensive and difficult [2, 3].

We propose a completely different approach to IC authentication in this paper. Our thesis is that there is enough manufacturing process variations across ICs with identical masks to uniquely characterize each IC, and this characterization can be performed with a large signal-to-noise ratio (SNR). The characterization of an IC involves the generation of a set of challenge-response pairs. To authenticate ICs we require the set of challenge-response pairs to be characteristic of each IC. For reliable authentication, we require that environmental variations and measurement errors do not produce so much noise that they hide inter-IC variations. We will show in this paper, using experiments and analysis, that we can perform reliable authentication using the techniques that we now introduce.

How can we produce a unique set of challenge-response pairs for each IC, even if the digital IC functionality or masks of the ICs are exactly the same? We rely on there being enough statistical delay variation for equivalent wires and devices across different ICs. Sources of statistical variation in manufacturing are well documented in the literature (e.g., [6] [5]) and statistical variation has been exploited to create IC identification circuits that generate a single unique response for each manufactured IC [10]. The *transient* response of the IC to a challenge, i.e., input stimulus, is de-

*This work was funded by Acer Inc., Delta Electronics Inc., HP Corp., NTT Inc., Nokia Research Center, and Philips Research under the MIT Project Oxygen partnership.

†Visiting researcher from Philips Research, Prof Holstlaan 4, Eindhoven, The Netherlands.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
CCS'02 November 18–22, 2002, Washington, DC, USA.
Copyright 2002 ACM 1-58113-612-9/02/0011 ...$5.00.

pendent on the delays of wires and devices within each IC. Our contribution is to show that by exploiting statistical delay variation and measuring transient response, one can generate multiple challenge-response pairs[1] that can be used to identify *and* authenticate an IC. The transient response only gives indirect information about the delays of wires and devices in the IC on the paths that are stimulated by the challenge. Since only indirect information is provided, it is possible to securely authenticate the IC.

To break the authentication methodology, the adversary can fabricate a "counterfeit" IC that produces exactly the same responses as the original IC for all challenges. Given the statistical variation inherent in any manufacturing process, we argue that the probability of this happening for a newly fabricated IC is very low, implying that the adversary will have to fabricate a huge number of ICs, and make comprehensive measurements on each one, in order to create and discover a counterfeit.

Alternately, the adversary can create a timing-accurate model of the original IC and simulate the model to respond to challenges, in effect creating a "virtual counterfeit." However, this model has to be extremely accurate since it has to incorporate near-exact delays of all devices and wires within the original IC, with errors of no more than 0.01%. Moreover, the transient response is a non-linear and non-monotonic function of the delays of wires and devices in the IC. The adversary has to invert this function to get the parameters of his model. We argue that this is very hard to do, even given complete mask information of the IC and unrestricted physical access to the IC. Further, we can make this even harder by restricting the challenges that can be presented to the IC and/or obfuscating the responses.

The rest of this paper will be structured as follows: In Section 2, we define PUFs. This is followed by an overview of our approach to creating silicon PUFs in Section 3. We describe various challenges in creating a silicon PUF in Section 4, and present an architecture for such a device. Then, we describe applications of silicon PUFs in Section 5. In Section 6 we describe preliminary experiments we have conducted using commodity FPGAs that indicate that there is enough statistical variation for authentication to be viable, and that give an idea of the difficulty of modeling or cloning silicon PUFs. Finally, we briefly discuss ongoing work in Section 7.

2. DEFINITIONS

DEFINITION 1. *A Physical Random Function (PUF)*[2] *is a function that maps challenges to responses, that is embodied by a physical device, and that verifies the following properties:*

1. *Easy to evaluate: The physical device is easily capable of evaluating the function in a short amount of time.*
2. *Hard to characterize: From a polynomial number of plausible physical measurements (in particular, determination of chosen challenge-response pairs), an attacker who no longer has the device, and who can only use a polynomial amount of resources (time, matter, etc...) can only extract a negligible amount of information about the response to a randomly chosen challenge.*

In the above definition, the terms short and polynomial are relative to the size of the device, which is the security parameter. In particular, short means linear or low degree polynomial. The term plausible is relative to the current state of the art in measurement techniques and is likely to change as improved methods are devised.

In previous literature [11] PUFs were referred to as Physical One Way Functions, and realized using 3-dimensional micro-structures and coherent radiation. We believe this terminology to be confusing because PUFs do not match the standard meaning of one way functions.

The focus of this paper is the silicon realization of PUFs, which we shall term silicon PUFs (SPUFs).

DEFINITION 2. *A type of PUF is said to be Manufacturer Resistant if it is technically impossible to produce two identical PUFs of this type given only a polynomial amount of resources.*

The silicon PUFs that we will describe in the sequel are manufacturer resistant, as they use circuit characteristics that are beyond the control of the fabrication process. When a PUF is manufacturer resistant, the amount of trust that must be placed in the manufacturer of the PUF is significantly reduced.

DEFINITION 3. *A PUF is said to be Controlled if it can only be accessed via an algorithm that is physically linked to the PUF in an inseparable way (i.e. any attempt to circumvent the algorithm will lead to the destruction of the PUF). In particular this algorithm can restrict the challenges that are presented to the PUF and can limit the information about responses that is given to the outside world.*

Silicon PUFs are ideally suited to being controlled PUFs. The PUF circuit can be intertwined with a circuit that controls access to the PUF in a very fine grained way. In [9], we go more into the details of controlled PUFs, how to use them, and the types of applications that they can support.

3. OVERVIEW OF APPROACH

We wish to implement a PUF in silicon so we can identify and authenticate a given integrated circuit (IC). By exploiting statistical variations in the delays of devices and wires within the IC, we create a manufacturer resistant PUF.

3.1 Manufacturing Variation

Manufactured ICs, from either the same lot or wafer have inherent delay variations. Across a die, device delays vary due to mask variations – this is sometimes called the system component of delay variation. There are also random variations in dies across a wafer, and from wafer to wafer due to, for instance, process temperature and pressure variations, during the various manufacturing steps. The magnitude of delay variation due to this random component can be 5%

[1] In fact, the number of potential challenge-response pairs grows exponentially with the number of inputs to the IC, since the response to each distinct challenge typically depends on a different set of device and wire delays within the IC. Of course these challenges are not all independent as a given circuit element will influence the response to many different challenges.

[2] PUF actually stands for Physical Unclonable Function. It has the advantage of being easier to pronounce, and it avoids confusion with Pseudo-Random Functions.

or more. Delay variations of the same wire or device in different dies have been modeled using Gaussian distributions and other probabilistic distributions (e.g., [6]). Constant research attempts to reduce all these sources of variation because they inherently limit the component density of the IC. Nevertheless, the relative variations in state of the art components tends to increase with time (see chapter 14 of [7]).

On-chip measurement of delays can be carried out with very high accuracy, and therefore the signal-to-noise ratio when delays of corresponding wires across two or more ICs are compared is quite high.

3.2 Environmental Variations

The most significant environmental condition that affects chip operation is ambient temperature. The delay of gates and wires depends on the junction temperature [13] which is dependent on the ambient temperature. Therefore, significant variations in the ambient temperature, e.g., ± 25 degrees Celsius, can cause appreciable variations in the delays. The main problem posed by this variation is the incorrect rejection of an authentic IC. However, relative measurement of delays, essentially using delay ratios, provides robustness against environmental variations, such as varying ambient temperature, and power supply variations. The impact of varying junction temperature can be reduced by using all the elements in the PUF in a uniform way. Our experiments in Section 6.1 validate the robustness of relative measurement.

For huge changes in environmental conditions, e.g., 100 degrees in ambient temperature, when even relative measurements break down, authentication can be carried out taking into account the existing environmental conditions. Essentially, a PUF would be seen as 2 or 3 different PUFs, only one of which is expressed at a time, depending on the temperature.

Finally, circuit aging can also change delays, but its effects are significantly smaller than temperature and power supply effects.

3.3 Challenge-Response Pairs

As we mentioned in the introduction, manufacturing variations have been exploited to identify individual ICs. However, the identification circuits used generate a *static* digital response (which is different for each IC). We propose the generation of many challenge-response pairs for each IC, where the challenge can be a digital (or possibly analog) input stimulus, and the response depends on the transient behavior of the IC, and can be a precise delay measure, or a digital response based on measured delay.

The transient behavior of the IC depends on the network of logic devices as well as the delays of the devices and interconnecting wires. Assuming the IC is combinational logic, an input pair $\langle v_1, v_2 \rangle$ produces a transient response at the outputs. Each input pair stimulates a potentially different set of paths in the IC. If we think of each input pair as being a challenge, the transient response of the IC will typically be different for each challenge.

The number of potential challenges grows with the size and number of inputs to the IC. Therefore, while two ICs may have a high probability of having the same response to a particular challenge, if we apply many challenges, then we can distinguish between the two ICs. More precisely, if the standard deviation of the measurement error is δ, and the standard deviation of inter-FPGA variation is σ, then for Gaussian distributions, the number of bits that can be extracted for one challenge is up to (though this limit is difficult to reach in practice):

$$\frac{1}{2} log_2(1 + \sigma/\delta)$$

By using multiple independent challenges, we can extract a large number of identification bits from an IC. Of course, the bits that are extracted for different challenges are not all independent. This is not a problem as only a few hundreds of bits are sufficient to identify a component. What is important is that the relation between bits that are extracted from different challenges be extremely hard to find and exploit.

Upon every successful authentication of a given IC, a set of challenge-response pairs is potentially revealed to an adversary. This means that the same challenge-response pair cannot be used again. If the adversary can learn the entire set of challenge-response pairs, he can create a model of a counterfeit IC. To implement this method, a database of challenge-response pairs has to be maintained by the entity that wishes to identify the IC. This database need only cover a small subset of all the possible challenge-response pairs. However it has to be kept secret as the security of the system only relies on the attacker not being able to predict which challenges will be made. If the database ever runs out of challenge-response pairs, it can be necessary to "recharge" it, by turning in the IC to the authority that performs the authentication.

With Controlled PUFs many of these limitations can be lifted. In particular, the reuse of a challenge-response pair can be considered, and "recharging" of a PUF can be done over an untrusted network. These improvements are detailed in [9].

3.4 Attacks

There are many possible attacks on PUFs – here, we look at four different types of attacks.

The adversary can attempt to duplicate a PUF by fabricating a counterfeit IC containing an identical PUF. However, even if the adversary has access to the masks of the IC, and unless the PUF is very simple, statistical variation will force the adversary to fabricate a huge number of ICs and precisely characterize each one, in order to create and discover a counterfeit. This is a very expensive proposition, both economically and computationally speaking.

Now assume that the adversary has unrestricted access to the IC containing the PUF. The adversary can attempt to create a model of the IC by measuring or otherwise determining very precisely the delays of each device and wire within the IC. Techniques like differential power analysis do not help much in determining precise delays of individual devices. Direct measurement of device delays requires the adversary to open the package of the IC, and remove several layers, such as field oxide and metal. Each of these layers has some effect on the delays of the underlying devices, and during this process, the delays of the devices will change. One can also design the package to have a significant effect on the delays of each device within the IC. Even in the case where the device can be opened without breaking the PUF, the adversary still has to probe it precisely. In doing that he runs the risk of changing delays because of coupling

between the circuit and his probe. Moreover, if he has to probe underlying wires, the adversary has to damage overlying wires. These wires actually can influence the delays of the underlying wires so the adversary once again runs the risk of breaking the PUF.

The adversary could try to build a model of the PUF by measuring the response of the PUF to a polynomial number of adaptively-chosen challenges.[3] We believe this to be the most plausible form of attack. However, we argue that there is a significant barrier to this form of attack as well (cf. Section 4.1 and Section 6.2). An important direction of research is to find a circuit that is provably hard to break by this method.

Finally, in the case of controlled PUFs, the adversary can attempt to attack the control algorithm that is attached to the PUF. This could be done by probing the control circuit to determine information that was supposed to be kept secret, or by attempting to override values in the control algorithm. We are currently studying ways to prevent this. Our most promising candidate is for the top layer of metal on the IC to be entirely occupied by PUF delay wires. Therefore, an adversary who tries to probe or drive underlying wires would have to damage an overlying PUF wire, which would change the PUF and make his efforts useless. The next step in our research consists in verifying these effects on real circuits.

4. ARCHITECTURE AND IMPLEMENTATION

This section covers some of the many challenges involved in creating a silicon PUF (SPUF). The architecture that is described here is a preliminary attempt to address the issues that are involved. We first describe characteristics required of a circuit so it can be used as a PUF taking into account security. We then present circuit implementations with varying complexity.

4.1 Security

Can the adversary, given the PUF f, implemented as a circuit C_f, find the delays of all internal wires and gates within C_f by applying a polynomial number of input challenges to C_f and measuring delays of C_f's paths? We will assume that he has detailed knowledge of the internal *structure* of C_f, and a good estimate of the delays of the gates and wires in C_f. The adversary can get this information from the mask layout of C_f, which is assumed to be public.

We will refer to both a gate or wire as a device in the sequel.

We first note that creating accurate timing models is an intensive area of research. Even the most detailed circuit models have a resolution that is significantly coarser than the resolution of reliable delay measurement. If an adversary is able to find a general method to attack silicon PUFs by determining polynomial-sized timing models that are accurate to within measurement errors, this would represent a breakthrough.

4.1.1 Linear Delay Models

If there exists an input vector pair such that under *arbitrary* delays in the circuit, an event propagates along a path P, then the path P is said to be *single event sensitizable* [8]. One way that the adversary can determine internal delays is if there is a set of paths in C_f that cover all the devices such that each path in the set is single event sensitizable. By assuming that the device delays that make up a path add up to the total path delay, the adversary can apply input stimuli and obtain an affine system of equations, relating measured path delays to device delays. These equations are such that a path delay is only dependent on the delays of devices that comprise the path. The number of equations is equal to the number of delay variables, which is linear in the size of C_f. Solving a linear system of equations in the continuous domain is easy, provided the determinant is non-singular[4].

However, this attack makes at least two assumptions, which are not necessarily true, as we show in Section 6.2. First, it assumes that the delays are additive, i.e., path delay is an exact sum of device delays. Second, it assumes that the delay of the path is only dependent on the delays of devices on the path. In reality the path delay may be dependent on the state of neighboring devices, which in turn depends on the challenge.

In order to confront the adversary with a greater barrier, we should ensure that a set of single event sensitizable paths as described above does not exist in the circuit implementation C_f of the PUF. Fortunately, most paths are not single event sensitizable – in fact, a careful structuring of logic is required to produce single event sensitizability [8].

4.1.2 Nonlinear Delay Models

What happens if multiple paths are actuated when an input stimulus is applied to C_f? Then, a much more complex set of equations will result[5]. Even if we assume device delays are additive, this system is not a linear system because:

- If two transitions of the same polarity ($0 \rightarrow 1$ or $1 \rightarrow 0$) arrive at a gate then the faster or slower one will go through depending on the type of gate. This means that the path delay is related to the maximum or minimum of two or more gate delays. For example, we may have:

  $$D(P_1) = MAX(g_1,\ MIN(g_2 + w_1, g_3 + w_2))$$

  where g_i is the gate delay of gate i and w_i is the wire delay of wire i. $D(P_1)$ is monotonic in the g_i's and the w_i's, but the set of equations is not necessarily separable, i.e., the adversary will not be able to write it in the form:

  $$g_i \ = \ F_i(g_1,\ \cdots, g_{i-1},\ g_{i+1}, \cdots,\ g_k,\ w_1, \cdots,\ w_l)$$

[3]Clearly, a model can be built by exhaustively enumerating all possible challenges, but this is intractable.

[4]The determinant is singular if the paths that were chosen are not independent. Choosing new paths that are independent should give new equations that will remove the singularity. If this does not help, then there are device delays in the circuit that never appear independently. These delays should be amalgamated into a single delay, as the attacker only needs the amalgamated delay for his model.

[5]In this section, we assume that we are measuring the delay between a change of input vector, and the response on the output of the circuit. It is important that the SPUF constrain the attacker to this model, by giving the circuit time to stabilize between consecutive changes of input stimulus. Otherwise, by very rapidly changing input stimuli, the attacker could try to determine which path is responsible for the delay of the circuit.

in order to easily solve it. (Note that some types of systems of nonlinear equations where the F_j are monotonic can be solved in polynomial time.)

- If two transitions of opposite polarity converge at a gate at different times, then the path delay can become a non-monotonic function of the gate delays. As a simple example, consider an AND gate where a rising transition arrives after a falling transition. In this case, the output of the AND gate is a constant 0, implying a path delay of 0. If the rising transition is sped up to arrive before the falling transition, the AND gate will glitch $0 \rightarrow 1 \rightarrow 0$, and the delay of the paths through the gate will become non-zero. Then, the relationships that the adversary has to write between the measured path delays and the device delays will become more complex.

Thus, to characterize a PUF the adversary has to solve a system of equations that are highly non-linear and non-separable.

4.1.3 Summary

Determining device delays by applying challenges to C_f requires the adversary to perform the tasks enumerated below.

- If the additive delay model is applicable, solve a non-linear, possibly non-separable and non-monotonic, system of equations that grows with the size of the PUF.
- If the additive delay model nearly applies, model device delays as being a function of the device's context (states of nearby devices) at the time of the challenge, which implies that the number of equations can grow significantly larger than the number of devices in the PUF.
- If the additive delay model does not apply at all, model path delays accurately as non-additive functions of device parameters. In general, the circuit analysis performed by tools such as SPICE [4] may be required to relate path delays to device parameters.

4.2 Circuit Implementation

Here we describe a straw-man implementation of a silicon PUF. In this implementation, we will measure the frequencies of parameterized self-oscillating circuits to characterize the IC that is being measured. In order to ensure robustness, we will measure delays through glitch-free circuits in which the total delay is a continuous function of the elementary device delays that make it up. Further, we will compensate for environmental variations by taking delay ratios. To improve security we will select circuits that exhibit non-monotonic behavior, i.e., for which the total delay is not a monotonic function of the elementary device and wire delays.

4.2.1 Structure of the self-oscillating circuit

Figure 1 is a simplified circuit that can be used to measure delays[6]. The delay circuit that is to be measured is placed in a self-oscillating circuit, the frequency of which is a function of the delay of the circuit. The resulting waveform is synchronized and its rising edges are counted by a counter. The counter is activated for a predefined number of clock cycles, after which the frequency of the self-oscillating loop can be read out of the counter. By placing many such loops on a chip, it is possible to measure many delays simultaneously. As we will see later, this plays an important part in compensating for variation of the measured frequency due to environmental variations.

For making an SPUF, the key is to find a circuit, the delay of which is a complicated function of the SPUF's input challenge, and that can be inserted in the self-oscillating loop.

4.2.2 A candidate delay circuit

Figure 2 shows a delay circuit with a number of attributes that are desirable for an SPUF delay circuit.

The circuit is made up of $n-1$ stages, where n is the number of bits in the challenge. Each stage is made up of two multiplexers (the trapezoids), and a few buffers (the triangles). If we ignore the buffers for now, what we have is a circuit with a top path and a bottom path. At the input to the delay circuit, a rising or falling edge gets sent into both the upper and lower path. At each stage of the circuit, depending on the value of the stage's challenge bit, the edges may cross, that is, the edge from the lower path goes to the higher path and vice versa. One of the two edges is then selected by the output multiplexer to be looped back to the input of the circuit in order for self-oscillations to occur.

The number of paths that can be measured this way is exponential in the number of stages in the delay circuit. However, the delays are clearly not independent, as there is a lot of sharing between paths. Worse, the path is sufficiently simple that an adversary could calculate the delays of the various parts of this circuit with only a linear number of measurements, if an additive delay model is assumed.

There isn't much that can be done about the dependence that exists between the paths, as the amount of variation that the delay function can exploit is only proportional to the size of the circuit. However, we can use strategies that make the dependence a lot more difficult to exploit by using the fancy variable delay buffers that appear in this circuit.

Indeed, that is what the buffers are used for in Figure 2. The buffers come in pairs, one of them is always on, while the other is only activated when the other path is low. This adds a complicated non-monotonic (if an elementary delay becomes longer, it is possible that the total delay will get shorter) interaction between the two edges that are racing through the circuit. Which prevents the attacker from simply writing a linear equation to get the delays of individual delay elements.

4.2.3 Compensated Measurement

Each of the circuits presented has a frequency counter that measures delays of paths. Since these delays are going to vary due to environmental conditions, it is crucial to compensate for these variations if we are to perform reliable identification or authentication. Compensation is carried out independent of the measurement during post-processing, simply by taking ratios of delays for different loops, or for different challenges on the same loop.

4.3 Improving a PUF Using Control

The PUF that we have described so far extracts informa-

[6] In order for the self-oscillating loop to function correctly, a more complicated circuit is often necessary to avoid problems with glitches in the delay circuit.

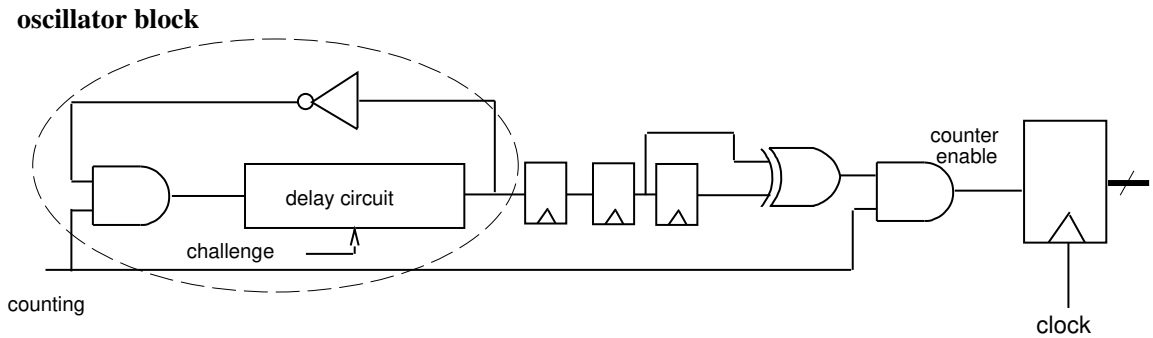


Figure 1: Self-Oscillating Loop Circuit.

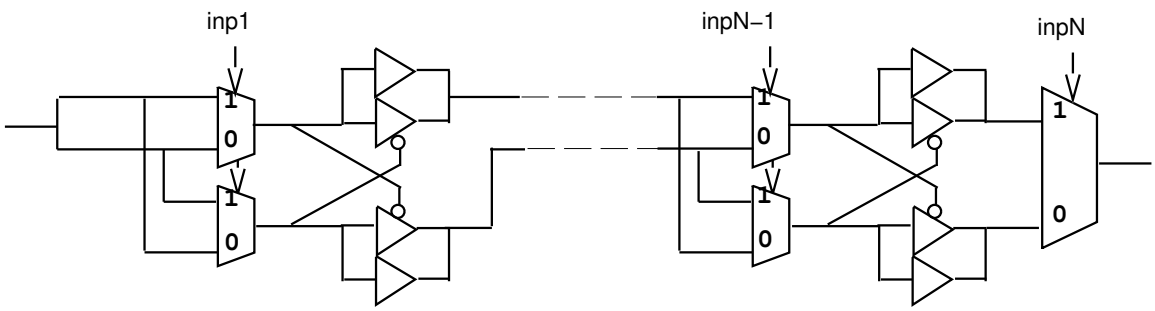


Figure 2: Non-Monotonic Delay Circuit.

tion that is component dependent from an IC, and it appears that this information is hard for an adversary to predict. By adding control to that PUF, it is possible to make it a lot stronger. This section describes some of the techniques that can be used to improve the reliability and strength of a PUF.

In each case, we have a PUF f that we are trying to improve in some way. Control allows us to improve f by constructing a new PUF g, that is based on f. The control only allows f to be evaluated as part of an evaluation of g, and only uses the result of the evaluation of f to help evaluate g.

The block diagram in figure 3 shows most of the improvements that are discussed in this section. The reader can refer to them to get a better understanding of what is being explained.

In this section we will be using random functions, a real implementation would naturally have to rely on pseudo-random functions.

4.3.1 Preventing Chosen Challenge Attacks

Unless one ventures into quantum effects (which would make a PUF highly unreliable), the number of physical parameters that define a PUF is proportional to the size of the system that defines it. Therefore, in principle, if an attacker is able to determine a number of primitive parameters that is proportional to the size of the physical system, he can use them to simulate the system and thus clone the PUF.

To try to determine primitive parameters, the attacker gets a number of challenge-response pairs (CRPs), and uses them to build a system of equations that he can try to solve. By definition, for a PUF, these equations are impossible to solve in reasonable time. However, there can be physical systems for which most CRPs lead to unsolvable equations, while a small subset of CRPs give equations that are able to break the PUF (which consequently is not really a PUF). Such a system is not secure because an adversary can use the CRPs that lead to simple equations to get a solvable system of equations, calculate the primitive parameters, and clone the PUF by building a simulator.

With control, it is nevertheless possible to build a secure system out of one of these broken PUFs. One way of doing this is for the control layer to simply refuse to give responses to challenges that lead to simple equations. Unfortunately, this method assumes that we know all the strategies that the attacker might use to get a simple set of equations from a chosen set of CRPs.

We can do even better if we pre-compose the broken PUF with a random function. Instead of using f directly, we use

$$g(x) = f(h(x)),$$

where h is a random function. With this method, it is impossible for the adversary to choose the challenge $h(x)$ that is being presented to the underlying PUF, so even if he finds a challenge that would break it, he is unable to present that challenge. Now, there is no need for the designer of the PUF to know what challenges the adversary might try to exploit.

4.3.2 Post-Composition with a Random Function

It is desirable for the output of a PUF to exhibit as much randomness as possible to prevent an adversary from guessing the response to one challenge by using the response to another challenge. However, the output of a physical sys-

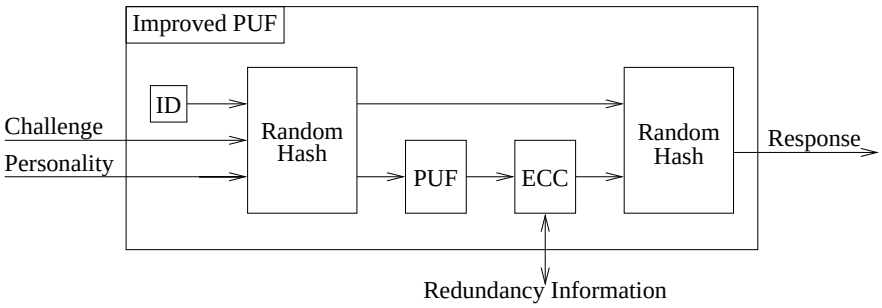


Figure 3: This diagram shows how control can be used to improve a PUF. Random hash functions are used at the input and output of the PUF, an Error Correcting Code is used to make the PUF reliable, a unique identifier guarantees that no two PUFs will be identical, and a personality selector allows the owner of the PUF to maintain his privacy.

tem is likely to produce similar responses when faced with similar stimuli. Moreover, as we discussed in section 4.3.1, CRPs can be used to get systems of equations that relate the PUF's underlying physical parameters.

Both of these risks can be eliminated by doing a simple transformation on the PUF. If f is the PUF that we are trying to improve, and h is a random hash function, then

$$g(x) = h(x, f(x))$$

is a stronger PUF. With this method, we can take a PUF that has good properties such as manufacturer resistance, and make it into a PUF that has the advantages of a digital PUF. The random hash function's avalanche-effect ensures that nearby outputs of f will lead to completely different outputs of the composite function, and the one-way[7] nature of h means that to set up a system of equations, the adversary has to invert h (or include the definition of h in the system of equations, which is just as bad).

Post-composing the PUF with a random function is a very important step because it makes the system provably resistant to non physical attacks, as long as enough information is extracted from the physical before running it through the output random function. In the case of a delay circuit, the right thing would be to measure a number of delays until a few hundreds of bits have been extracted from the system, and then run the lot of them through the random function.

4.3.3 Giving a PUF Multiple Personalities

A possible concern with the use of PUFs is in the area of privacy. Indeed, past experience shows that users feel uncomfortable with processors that have unique identifiers, because they feel that they can be tracked. PUFs being a form of unique identifier, users could have the same type of concern with their use.

This problem can be solved by providing a PUF with multiple personalities. The owner of the PUF has a parameter that she can control that allows her to show different facets of her PUF to different applications. To do this, we hash the challenge with a user-selected personality number, and use that hash as the input to the rest of the PUF.

In this way, the owner effectively has many different PUFs at her disposal, so third parties to which she has shown different personalities cannot determine if they interacted with the same PUF.

We go into the details of protocols that use multiple personalities in [9].

4.3.4 Error Correction

In many cases, the PUF is being calculated using an analog physical system. It is inevitable that slight variations from one run to the next will cause slight changes in the digitized output of the PUF. This means that the chip only produces an approximation of the response that is expected of it. In some applications, the chip and the challenger cannot directly compare the real response with the desired response as this would require sending one of the responses in the clear, thus compromising the shared secret. Therefore, something must be done to make the PUF's output identical each time a challenge is reused.

A suitably selected error correcting code is one possibility. When a challenge-response pair is created, some redundant information is also produced that should allow slight variations in the measured parameters to be corrected for. On subsequent uses of the challenge-response pair, the redundant information is provided to the PUF along with the challenge. It is used to correct the response from the physical system.

Naturally, the error correction must take place directly on the measured physical parameters. In particular, if the PUF is post-composed with a random function, the correction must take first. If multiple measurements are being combined into one response, the error correction should operate on all the measurements.

It is of course critical that the redundancy information not give away all the bits of the response.

4.3.5 Multiple Rounds

To add even more complexity to the attacker's problem, it would be possible to use the PUF circuit multiple times to produce one response. The corrected response from one round can be fed back into the PUF circuit. After a few rounds have been done, all their outputs could get merged together along with the challenge, the personality and the chip's identifier and passed through a random hash function to produce the global response.

4.3.6 Unique Identifier

[7] Random functions are one-way functions.

With manufacturer resistant PUFs, the manufacturer resistance is typically a result of the manufacturer's limited control over process variations. Each PUF is different because of these variations. However, it is possible that there will be identical PUFs. This isn't much of a problem, because in general finding a pair of PUFs that is identical requires producing, and comparing an unreasonable number of PUFs.

Nevertheless, it is possible to guarantee that any two PUFs are different. To do so, we combine the actual challenge and a unique identifier that is unique to the chip with a hash before running them through the rest of the PUF. The unique identifier that is used here need not be secret, and can be the IC's serial number, for example.

In this way, no two PUFs are identical, and even if two CPUFs share the same underlying PUF f, there is no way for an adversary to find this out (the manufacturer might be able to discover it before setting the PUF's unique identifier, but the cost of testing is prohibitive in any case).

5. APPLICATIONS

What are the benefits of having a unique hardware chip? We believe there are many, and we describe a few applications here. Other applications can be imagined by studying the literature on secure coprocessors. In particular, [15] describes many applications that this work should be applicable to. The authenticated identification application that is listed applies to PUFs in general. It is in fact the only application of PUFs until control is added. The other applications require controlled PUFs in order to be possible, the relevant theory can be found in [9]. The important point is that with control, it is possible for a PUF to be used to provide a shared secret to an application.

5.1 Authenticated identification

The easiest application to implement is authenticated identification. It is the application that was described in [11]. One possible application is to securely identify smartcards. We can create a smartcard with a PUF, and each time the PUF-smartcard is used, the card reader can ask the card for responses to a specific set of challenges to identify the PUF. In this case each time the PUF-smartcard is used, a new set of challenges has to be used, else the PUF-smartcard is subject to replay attacks. This does not pose a problem, since the card manufacturer can create a large number of challenge-response pairs before the PUF-smartcard is given to a user.

With current methods, it is possible for someone who is in possession of a smartcard to produce a clone of it, by extracting its key information through one of many well documented attacks. If someone loses track of her card for a while, her card can potentially have been cloned. Being in physical possession of the smartcard is therefore not synonymous to being safe. With a PUF on the smartcard that can be authenticated and identified, there is no longer any need for a digital key that can be easily extracted. The smartcard hardware is itself the secret key. This key cannot be duplicated, so a person can lose control of the PUF-smartcard, retrieve it, and continue using it. In this way it is possible to lend the PUF-smartcard to a "friend" without causing a permanent breach of security.

This method is well suited to credit cards since the important point is to check that the person is in posession of her original card. It does not, however provide guarantees that the card reader is really talking to the original card, as it is possible that a man in the middle attack is being carried out. To get around this limitation for more sophisticated applications requires control and the protocols described in [9].

In section 6, we show that with 10 self-oscillating loops such as those we have studied, it is possible to distinguish between up to 10 billion chips. In the same conditions, an adversary who tries to guess the response correctly would have only one chance in 10^20 billion of succeeding. This number need not be any greater because the adversary will exhaust the prerecorded database of challenge-response pairs long before he gets a significant probability of success.

In this case, the adversary will, however have successfully carried out a denial of service attack. This attack can be made as hard as breaking a non-PUF system by requiring that the smartcard identify itself using a digital challenge-response protocol before it challenges the card with one of the limited number of PUF challenge-responses that it has.

Note that this method, only allows authentication of the smartcard to a remote server. It does not remove the need for a PIN number, or biometrics, or some other means for the card to identify the bearer of the card.

5.2 Proof of Execution on a Specific Processor

At present, computation power is a commodity that undergoes massive waste. Most computer users only use a fraction of their computer's processing power, though they use it in a bursty way, which justifies the constant demand for higher performance. A number of organizations, such as SETI@home and distributed.net, are trying to tap that wasted computing power to carry out large computations in a highly distributed way. This style of computation is unreliable, however, as the person requesting the computation has no way of knowing that it was executed without any tampering.

With chip authentication, it would be possible for a certificate to be produced that proves that a specific computation was carried out on a specific chip. The person requesting the computation can then rely on the trustworthiness of the chip manufacturer who can vouch that he produced the chip, instead of relying on the owner of the chip.

There are two ways in which the system could be used. Either the computation is done directly on the secure chip, or it is done on a faster insecure chip that is being monitored in a highly interactive way by supervisory code on the secure chip ([15]).

5.3 Code that Runs Only on a Specific Processor

The software industry is always looking for ways to limit the use of its products. We are exploring ways in which a piece of code could be made to run only on a processor with a PUF. In this way, pirated code would fail to run. One method that we are considering is to encrypt the code using the PUF's challenge-response pairs on an instruction per instruction basis.

6. EXPERIMENTS

To date, a number of experiments have been conducted using Xilinx XC2S200 Field Programmable Gate Arrays (FPGAs). The results to date are preliminary, but provide evidence that silicon PUFs can be used to perform reliable

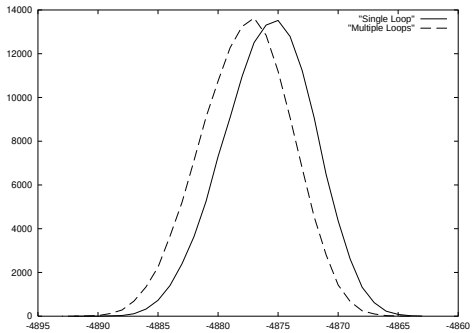


Figure 4: In this plot we show how multiple self-oscillating loops on the same IC interfere. A loop's frequency was determined first when that loop was oscillating alone, and second when the seven other loops on the chip were turned on. As can be seen the change in frequency between these two situations is tiny compared with measurement noise. This suggests that the interference between the loop and other parts of the IC is minimal and can be ignored.

authentication, and that it is hard for the adversary to create a timing model of the PUF.

6.1 Usable Process Variability is Present

FPGAs are an example of a high-volume part where the manufacturing process is tuned to produce ICs that are as identical as possible in order to maximize yield and performance. Our experiments indicate that even a highly-optimized manufacturing process designed for predictability has enough variability to enable reliable identification.

In all our experiments, we compare delays across two or more FPGAs with each FPGA being programmed by exactly the same personality matrix. This means that each FPGA has exactly the same logic circuit, and moreover the circuit is implemented in FPGA modules in the exact same locations. Therefore, these FPGAs can be viewed as copies of the same IC.

In our first experiment each FPGA is equipped with 8 self-oscillating loops, the circuit for which is shown in Figure 1. Each loop is made up of 32 buffers[8] and an inverter. We determine the frequencies of the loops by measuring the number of oscillations they make during a certain period of time (typically 2^{20} cycles of an external 50 MHz oscillator). The period of the loops is on the order of $60ns$.

We ran experiments to quantify measurement errors, inter-FPGA variation, variation due to ambient temperature and variation due to power supply voltage variations. To summarize our findings, the following standard deviations are given in parts per million (ppm). A deviation of n ppm around a frequency f_0 corresponds to a deviation of $\frac{n f_0}{10^6}$. These deviations correspond to measurement across several FPGAs.

1. Consecutive measurements of the same delay produce slightly different results because of measurement inaccuracy inherent in the loop circuit circled in Figure 1. The standard deviation of this measurement error with compensated measurement is 30 ppm.

2. The standard deviation in inter-FPGA delays with compensated measurements is from $5000ppm$ to $30000ppm$ depending on the pair of loops that is used for the measurement. Figure 6 shows an example of the relationship between measurement error and inter-FPGA variation for four different FPGAs. Clearly identification information can be extracted from the frequencies of the loops that we are measuring.

3. The frequency of a loop can be influenced by nearby circuitry. To try to evaluate the magnitude of this interference we compared the frequency of one of the loops when the other loops on the FPGA were turned on or off. The deviation we observed was $10ppm$. Figure 4 shows the frequency distribution for a loop when the other loops are turned on or off.

4. The variation in frequency when the ambient temperature is varied from 25 to 50 degrees Celsius is $50000ppm$ for uncompensated measurements. This is sufficient to prevent FPGA identification. Fortunately, compensation (see 4.2.3) reduces this to $100ppm$. Figure 7 illustrates the temperature dependence with and without compensation.

5. Power supply voltage variations are also compensated to a large extent using our scheme. Around the FPGA's 2.5V operating point, the variation of the compensated measurement with voltage is about $3000ppm/V$ as shown in Figure 5. In practice external power supply variations can be kept to within 1%, which corresponds to $1\% \times 2.5V \times 3000ppm/V = 75ppm$. Therefore, commonly available voltage regulators will suffice to keep the supply voltage within tolerable bounds. It is interesting to note that the compensated measurement seems to have an extremum around $2.7V$. By running the FPGAs at $2.7V$ instead of $2.5V$ this extremum could be used to further improve the robustness of the measurements.

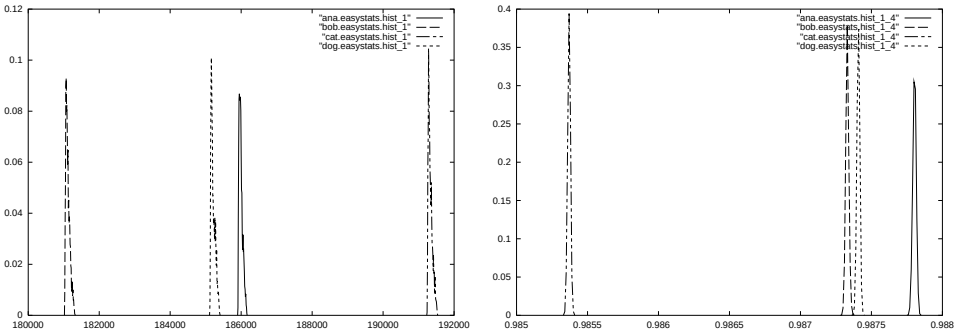
6. Circuit aging can create variance in measurements carried out over a long period of time. However, the effect of circuit aging is typically significantly less than power supply or temperature variation. Future study will have to check the impact of aging on the measurements.

Given the numbers above, if we take 100ppm as a rough estimate of the noise, and 10000ppm as a rough estimate of the signal, then we have a signal to noise ratio of 100. If the noise distribution was Gaussian (this is not really the case as some parts of the noise are due to slowly varying parameters such as Temperature and supply voltage), we would be able to extract 3.3 bits per measurement. So with 10 measurements, done on 10 different loops, we could distinguish between 10 billion different chips.

To summarize the experiments in this section, compensated measurements enable reliable identification under appreciable environmental variations.

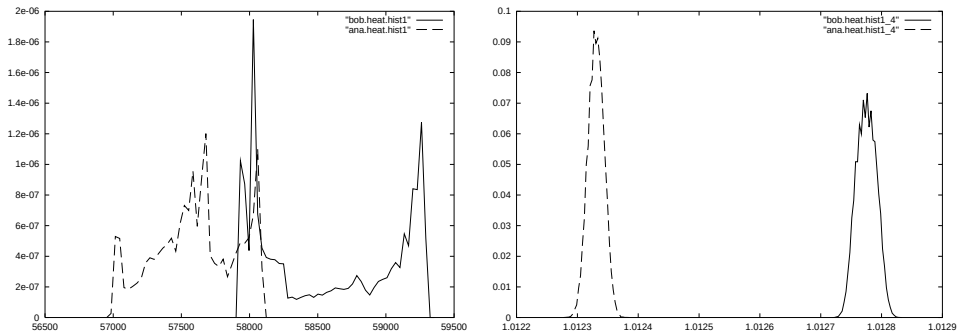
We note that variance in a manufacturing process can be increased quite easily by making small changes in the fabrication steps, e.g., not regulating temperature and pressure

[8] In this context, a buffer is simply a logic gate that copies its input to its output with a short delay.

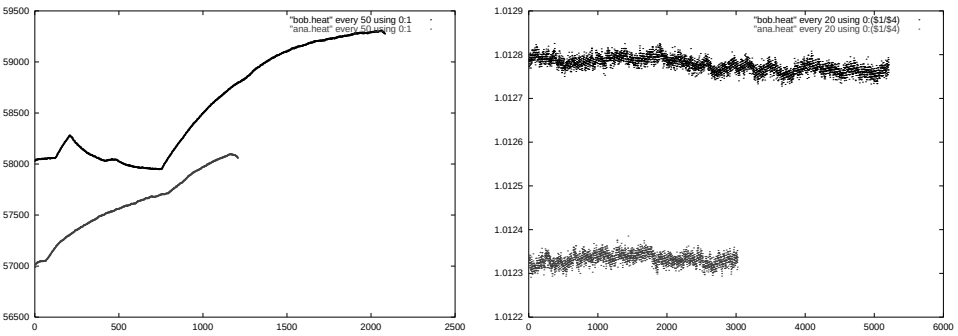


(a) Uncompensated (b) Compensated

Figure 6: These histograms show the relation between measurement error (width of a peak) and inter-FPGA variation (each peak is for a different FPGA), with and without compensation. Clearly information about the FPGA's identity can be extracted from these measurements.



(a) Uncompensated (b) Compensated



(c) Uncompensated (d) Compensated

Figure 7: These graphs show the results of an experiment in which two FPGAs had their ambient temperature vary between $25^{\circ}C$ and $50^{\circ}C$. The top plots show the measurement value versus time (in half-second sampling intervals). Note that the two FPGAs did not undergo the same temperature changes at the same time. The bottom plots are histograms of the respective plots on top.

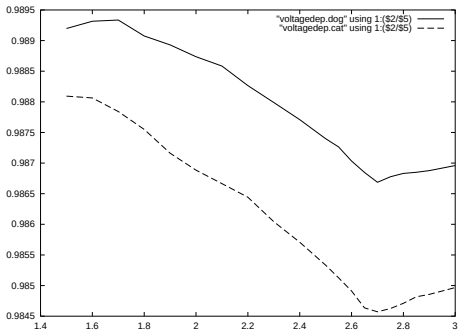


Figure 5: This plot shows compensated measurement dependency on power supply voltage. The dependency for 1% changes in supply voltage is small enough for our purposes. Interestingly, by running the FPGAs near the $2.7V$ extremum, it might be possible to further reduce the voltage dependency.

as tightly, and increased variance will allow reliable identification under a greater amount of environmental variation. Also, with the advent of deep submicron (e.g., 90 nm) devices, there is greater intrinsic fluctuation for minimum width devices due to lithography tolerance and dopant fluctuation [14]. Finally, an IC containing a PUF could be placed in an environment-resistant board to improve reliability.

6.2 How hard is model building?

We ran the same experiments on the (single event sensitizable) demultiplexer circuit shown in Figure 8. A circuit with 12 stages was used in our experiments.

The observed measurement error, inter-FPGA variation and dependence on environmental conditions were compatible with the results from section 6.1.

In addition to confirming the results from the previous experiments, the new circuit was able to show us the effect of challenges on the frequency of the self-oscillating loops. Figure 9 shows the compensated response of two different FPGAs as a function of the input challenge.

There is a clear dependency of the output on the challenge. Moreover, and quite predictably, there is a lot of structure in the challenge-dependence of the response. This structure is common to the two FPGAs and is due to large differences between paths in given stages of the delay circuit. To actually see a difference between the two FPGAs, one must look at the small scale differences between the two plots (we are looking for 1% variations on a plot that covers 50% variations). These differences are present, and appear most clearly as a difference in texture between the plots for the two chips.

The reason why such a simple circuit was chosen for this experiment is that we wanted to quantify how well an adversary could simulate the circuit by choosing an additive delay model. Indeed, suppose that the adversary wanted to create a model for the demultiplexer circuit of Figure 8. He reasons that the delay of the circuit under each challenge is the delay of the actuated path for that challenge. He can assume as additive delay model, where the delay of a path is the sum of the delays of the devices and wires on that path. By measuring the delay of a set of paths that cover all the devices and wires in the circuit, he can set up a linear system of equations that relate the unknown device and wire delays to known path delays. He can then solve for the device and wire delays, thereby obtaining a model of the circuit, which he can then simulate to guess at the response for an arbitrary challenge. The question then is: "How accurate is the model created by the adversary?" If the model is inaccurate, then the adversary can try to augment it by adding non-additive delay behavior or additional variables, and continue. The effort involved in non-additive model building is considerably higher but also difficult to quantify. Here, we will restrict ourselves to quantifying the complexity/error tradeoff of additive model building.

To quantify the accuracy of an additive model that the adversary can build, we measured the delays of all 2^n paths in a $n = 12$-stage demultiplexer circuit. Each of these paths corresponds to a different challenge. For a pair of paths P_1 and P_2 whose challenges differ in exactly one bit, the paths share all but one device. The adversary may assume an additive delay model which implies that the relationship between the path delays is

$$P_1 \; - \; P_2 \; = \; d_i \; - \; d_j \; .$$

The d_i and d_j pairs are marked on Figure 8.

Using all 2^n measured delays, we determined a mean and standard deviation for each of the $d_i - d_j$ quantities. This standard deviation is characteristic of the inaccuracy of the additive model, we shall call it σ_{calc}. In our experiments σ_{calc} was between $5ppm$ and $30ppm$, which is roughly the same as the environmental variations that we have to deal with. Thus, the additive model might be a valid way of breaking simple circuits such as single event sensitizable circuit of Figure 8.

Nevertheless, even if the additive delay model gives results that are within the tolerances that the adversary has to meet, he may not be able to use it to efficiently simulate the circuit. Indeed, when he uses the additive delay model, the adversary is essentially starting from a challenge he knows a response to, and performing a certain number of modification steps to the corresponding delay to account for differences between the known challenge and the one he is trying to calculate the response for. The modeling error, σ_{calc} is present for each one of the additions that the adversary performs. It is likely that the error that is committed when the model is applied multiple times will be greater than the best-case error that we have evaluated.

For example, if we assume that the errors that the adversary commits at each step of his computation are Gaussian and independently distributed between steps, then for a k step computation, the adversary in fact commits an error of $\sqrt{k}\sigma_{calc}$. The number of measurements that the adversary would have to make to be able to predict the response to a randomly selected response in fewer than k steps is exponential in $\frac{n}{k}$, so for big enough n, the additive delay model attack will not be sufficient even for simple circuits.

The use of circuits such as the variable delay buffer circuit of Figure 2 precludes an additive model based attack, since the delays are non-additive functions of the challenge.

7. ONGOING AND FUTURE WORK

There is still much to be studied about silicon PUFs.

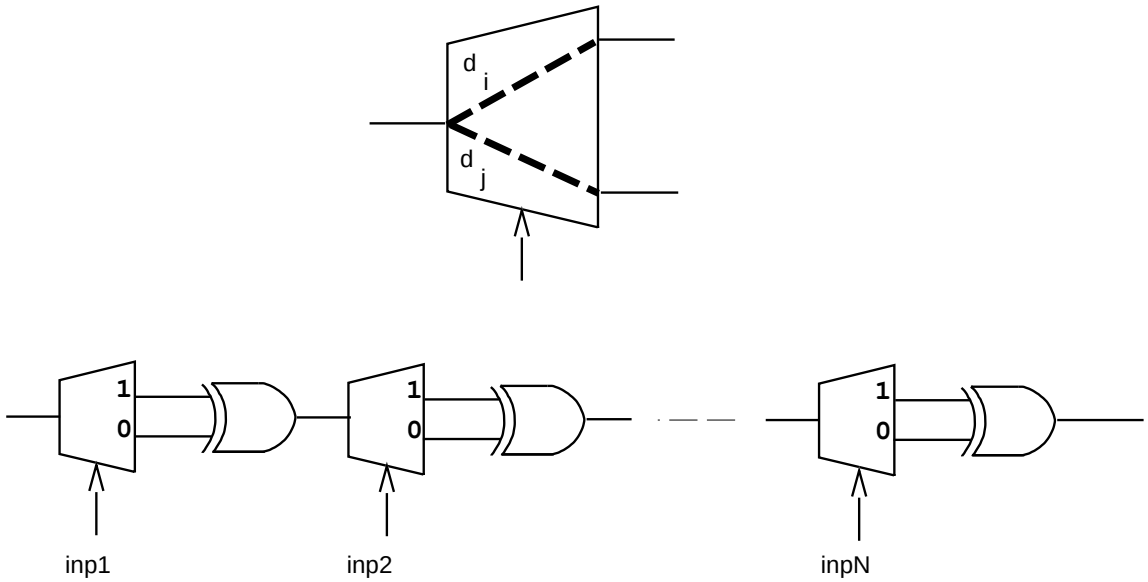
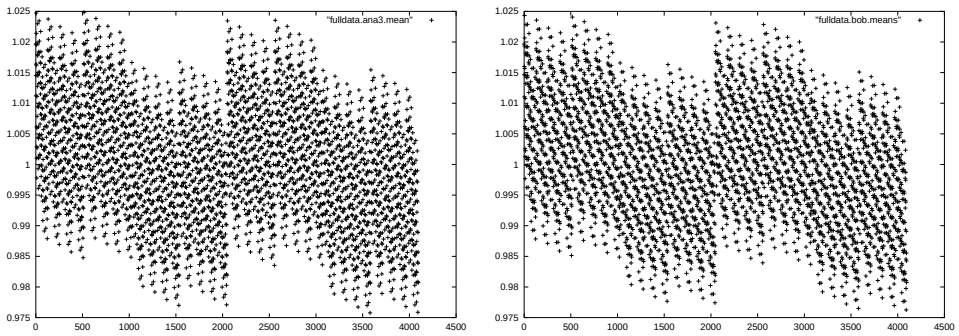


Figure 8: The demultiplexer circuit, used to test the feasibility of additive delay modeling of a PUF circuit.



(a) FPGA 1 (b) FPGA 2

Figure 9: Compensated Delay versus Input Challenges for the Demultiplexer circuit on two different FPGAs: The large scale structure is identical, and is due to differences in routing of paths on a given circuit. The difference between the FPGA appears at a much smaller scale, and can be seen as a difference in texture between the two plots.

First of all, it would be very satisfying to base the security of a silicon PUF on some previously known hard problem. One approach that we are considering, is to use a known pseudo-random function (PRF), used out of its normal operating conditions as the PUF circuit. In that case, it might be possible to relate the security of the PUF to the security of the PRF.

It would also be good to find better ways of measuring physical characteristics of the chip. Measuring delays directly instead of using self-oscillating circuits would allow the silicon PUF to operate much faster, allowing it to be used in protocols that require large numbers of uses. Improved measurement techniques might also make it possible to use circuits with glitches to reliably extract information about the chip. These circuits would be harder to simulate, making the adversary's problem harder. Moreover, in the case of Controlled PUFs, it is conceivable that differential power analysis techniques could be used on self-oscillating circuits to read challenges off the PUF against its will. The use of direct delay measurement should greatly reduce the signature of the delay measurement on the IC's power supply.

Another great improvement would be to find a way to use any sufficiently complex circuit, and suitably instrumented circuit as a SPUF. This would make the cost of adding PUF support to a circuit very low, and would guarantee that the PUF is inseparable from the circuit that it is supposed to accompany. This is particularly important in the case of CPUFs.

Finally, a detailed study of the physical attacks that the adversary can carry out is necessary. In particular it is important to know if probing the PUF circuitry using advanced non-invasive techniques can help build a simulation model of the PUF, and if so the physical barriers that can be placed against such probing must be considered.

8. CONCLUSION

We have described the notion of a Physical Random Function (PUF) and shown that a silicon PUF can be created.

The obvious application of a silicon PUF is authentication. Authentication has to be carried out reliably, minimizing the likelihood of false positives or false negatives. In order to perform reliable authentication, we proposed a circuit architecture for a PUF where delays are measured relative to other delays. This lends robustness to our scheme, and preliminary experiments indicate that authentication can be carried out reliably under significant variations in environmental conditions. To be robust against more significant environmental variations, careful circuit and package design is required. Fortunately, the VLSI design community is already addressing these problems in the realm of high-performance circuit design. In addition, a manufacturing process that produces high variations in device delays will result in higher signal-to-noise ratios and enable improved reliability.

The most plausible attack on a PUF is the model-building attack, where an adversary has access to the packaged IC containing the PUF, and can apply arbitrary challenges and monitor the resulting response. We have presented a preliminary analysis of this problem and our experiments indicate model-building is hard due to the precision requirements, but more work needs to be done in both analysis and experimentation.

While many problems remain to make PUFs useful and practical, we believe that this is a very promising low cost approach to improving the physical security of devices, especially when it is coupled with the ideas on controlled PUFs from [9].

9. REFERENCES

[1] R. J. Anderson. *Security Engineering: A Guide to Building Dependable Distributed Systems.* John Wiley and Sons, 2001.

[2] R. J. Anderson and M. G. Kuhn. Tamper Resistance – A Cautionary Note. In *Proceedings of Second Usenix Workshop on Electronic Commerce*, pages 1–11, 1996.

[3] R. J. Anderson and M. G. Kuhn. Low Cost Attacks on Tamper Resistant Devices. In *Proceedings of the* 5th *Security Protocols Workshop, Lecture Notes in Computer Science 1361*, pages 125–136. Springer-Verlag, Berlin, 1998.

[4] P. Antognetti and G. Massobrio. *Semiconductor Device Modeling with SPICE.* McGraw Hill, 1988.

[5] K. Bernstein. *High Speed CMOS Design Styles.* Kluwer Academic Publishers, 1998.

[6] D. S. Boning and S. Nassif. Models of Process Variations in Device and Interconnect. In A. Chandrakasan, W. Bowhill, and F. Fox, editors, *Design of High Performance Microprocessor Circuits*, chapter 6. IEEE Press, 2000.

[7] D. Chinnery and K. Keutzer. *Closing the Gap Between ASIC & Custom.* Kulwer Academic Publishers, 2002.

[8] S. Devadas, A. Ghosh, and K. Keutzer. *Logic Synthesis.* McGraw Hill, New York, NY, 1994.

[9] B. Gassend, D. Clarke, M. van Dijk, and S. Devadas. Controlled Physical Random Functions. In *Proceedings of the 18th Annual Computer Security Conference*, December 2002.

[10] K. Lofstrom, W. R. Daasch, and D. Taylor. IC Identification Circuit Using Device Mismatch. In *Proceedings of ISSCC 2000*, pages 372–373, February 2000.

[11] P. S. Ravikanth. *Physical One-Way Functions.* PhD thesis, Massachusetts Institute of Technology, 2001.

[12] S. W. Smith and S. H. Weingart. Building a High-Performance, Programmable Secure Coprocessor. In *Computer Networks (Special Issue on Computer Network Security)*, volume 31, pages 831–860, April 1999.

[13] N. Weste and K. Eshraghian. *Principles of CMOS VLSI Design: A Systems Perspective.* Addison Wesley, 1985.

[14] H. Wong and Y. Taur. Three-dimensional atomistic simulation of discrete random dopant distribution effects in sub-0.1 um MOSFETs. In *IEDM Technical Digest*, pages 705–708, 1993.

[15] B. S. Yee. *Using Secure Coprocessors.* PhD thesis, Carnegie Mellon University, 1994.