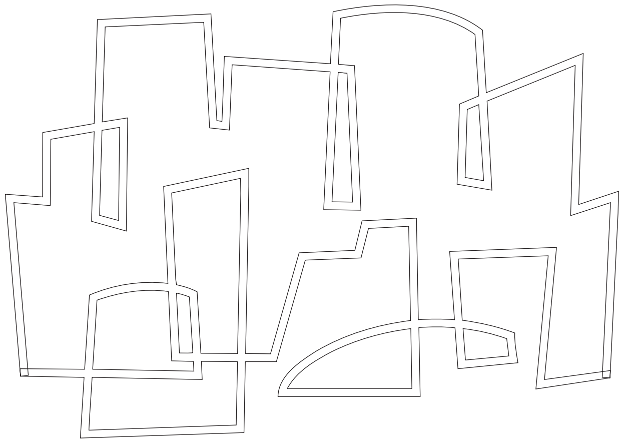
Delay-Based Circuit Authentication and Applications

Blaise Gassend, Dwaine Clarke,
Marten van Dijk, Srini Devadas

In the proceedings of the 18th Symposium on
Applied Computing, March 2003

2002, October

Computation Structures Group
Memo 455



Delay-Based Circuit Authentication and Applications

Blaise Gassend Dwaine Clarke Marten van Dijk Srinivas Devadas

Laboratory for Computer Science
Massachusetts Institute of Technology

October 24, 2002

Abstract

We describe a technique to reliably identify individual integrated circuits (ICs), based on a prior delay characterization of the IC.

We describe a circuit architecture for a key card for which authentication is delay based, rather than based on a digital secret key. We argue that key cards built in this fashion are resistant to many known kinds of attacks.

Since the delay of ICs can vary with environmental conditions such as temperature, we develop compensation schemes and show experimentally that reliable authentication can be performed in the presence of significant environmental variations.

The delay information that is extracted from the IC can also be used to generate keys for use in classical cryptographic primitives. Applications that rely on these keys for security would consequently be less vulnerable to physical attack.

Introduction

We describe a technique to identify and authenticate arbitrary integrated circuits (IC's) based on a prior *delay* characterization of the IC. While IC's can be reliably mass-manufactured to have identical digital logic functionality, the premise of our approach is that each IC is unique in its delay characteristics due to inherent variations in manufacturing across different dies, wafers, and processes. While digital logic functionality relies on timing constraints being met, different ICs with the exact same digital functionality will have unique behaviors when these constraints are not met, because their delay characteristics are different.

Researchers have proposed the addition of specific circuits that produce unique responses due to manufacturing variations in IC's such that these IC's can be identified (e.g., [LDT00]). However, with these techniques, the focus is simply on assigning a unique identifier to each chip, without having security in mind. In order to authenticate an IC, a key has to be placed within the IC, access to the key has to be restricted to cryptographic primitives, and the IC has to be made tamper resistant, so an adversary cannot determine the key without destroying it. In essence, digital information has to be hidden in the IC. This information can then be used to simply identify the IC, or it can be used to enable a wide range of applications that rely on keyed cryptographic primitives.

Making an IC tamper-resistant to all forms of attacks is a challenging problem and is receiving some attention [And01]. 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 [KJJ99], that attempts to determine the key by stimulating the IC and observing the power and ground rails. 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 [SW99]. 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 [AK96, AK98].

We propose that authentication be based on hidden *delay* or timing information corresponding to a circuit rather than digital information. We will argue that the level of tamper resistance required to hide delay information is significantly less than for digital information. Invasive methods to determine device and wire delays will invariably change the delay of the devices or wires upon removal of the package or metal layers. Further, non-invasive attacks that are sometimes successful in discovering secret digital keys such as differential power analysis (DPA) [KJJ99] and electromagnetic analysis (EMA) [QS01] fail to provide precise enough delay information to break delay-based authentication. An important difference between hiding digital information versus timing information is that in the former case the manufacturer can produce many ICs with the same hidden digital key, but it is very hard, if not impossible, for a manufacturer to produce two ICs that are identical in terms of their delay characteristics.

To elaborate, 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.

The rest of this paper will be structured as follows: An overview of our approach to identify and authenticate ICs based on delays is given in Section 2. In section 3 we describe some applications, in particular a secure key card application. We describe the notion of a physical random function, which is what we are trying to implement, in Section 1. We argue that a particular circuit can be viewed as a physical random function and is resistant to various types of attacks in Section 4. In Section 6 we describe experiments we have conducted using commodity FPGAs that indicate that there is enough statistical variation for authentication to be viable, and that authentication can be carried out in a reliable manner using compensated measurements.[1]

1 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 predict: 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 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 [Rav01] 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 [MvOV96].

[1] A preliminary set of experiments for a simpler circuit are presented in [GCvDD02b].

[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.

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 (time, money, silicon, etc.).*

Manufacturer resistant PUFs are the most interesting form of PUF as they can be used to make unclonable systems.

We will describe how we can create silicon PUFs using delay characterization in the next section. We will argue in subsequent sections that it is hard to completely characterize the timing/delay of silicon PUFs.

2 Delay-Based Authentication

Our approach and the reasoning behind it is summarized in the next three subsections.

2.1 Statistical Delay Variation

When a circuit is replicated across dies or across wafers, manufacturing variations cause appreciable differences in circuit delays. 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% or more for metal wires, and is higher for devices (see chapter 12 of [CK02]). Delay variations of the same wire or device in different dies have been modeled using Gaussian distributions and other probabilistic distributions (e.g., [BN00], [Ber98]).

We briefly note here that in our experiments, the standard deviation of path delays in our example circuits across different FPGAs was in the range of 400 ppm.

2.2 Environmental Effects

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, provided environmental variation is low. To keep the signal-to-noise ratio high under significant environmental variations, we require compensated delay measurement (cf. Section 6). Using compensated delay measurement, under significant temperature and power supply variation[3], we can keep the standard deviation of compensated delays to within 25 ppm, which is significantly smaller than the standard deviation of inter-chip variation.

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

2.3 Generating Challenge-Response Pairs

As we mentioned in the introduction, manufacturing variations have been exploited to identify individual ICs. However, the identification circuits used thus far 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, a delay ratio, or a digital response based on measured delay or ratios.

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

[3]Temperature and power supply voltage have a significant affect on the absolute values of circuit delays [WE85].

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 huge number of identification bits from an IC. Actually producing a large number of bits is difficult to do in practice with multiple challenges because the responses to challenges are not independent. However, it is much easier to extract the information from the measurements if we are willing to get less than the maximum number of bits, and in the case where $\delta << \sigma$.

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 may be necessary to "recharge" it, by turning in the IC to the authority that performs the authentication.

3 Applications

3.1 Secure Keycard

The simplest application for PUFs is to make tamper-resistant, unforgeable key cards. This application was first described in [Rav01]. We will argue in Section 4 that silicon PUFs are difficult to forge and, as a result, these key cards are difficult to clone. The cards can also be combined with biometrics to help identify users.

These cards can be used for authenticated identification, in which someone or something with physical access to the card can use it to gain access to a protected resource. The general model is that of a principal with the key card presenting it to a terminal at a locked door. The terminal can connect via a private, authentic channel to a remote, trusted server. The server has already established a private list of Challenge-Response Pairs (CRPs) with the card. When the principal presents the card to the terminal, the terminal contacts the server using the secure channel, and the server replies with the challenge of a randomly chosen CRP in its list. The terminal forwards the challenge to the card, which determines the response. The response is sent to the terminal and forwarded to the server via the secure channel. The server checks that the response matches what it expected, and, if it does, sends an acknowledgment to the terminal. The terminal then unlocks the door, allowing the user to access the protected resource. The server should only use each challenge once, to prevent replay attacks; thus, the user is required to securely renew the list of CRPs on the server periodically.

3.2 Controlled PUFs

As we have implemented them in this paper, card-PUFs can be used for authenticated identification, as described above. However, unlike the PUFs from [Rav01], silicon PUFs can be accompanied on the same chip with control logic that restricts access to the PUF. In this case we have a Controlled PUF. By using the methods described in [GCvDD02a] a controlled PUF can be used to establish a shared secret between a remote party and a trusted chip. Because of the way the secret is embedded in the PUF, it is much harder for an adversary to impersonate the trusted chip than it would be if the chip had a secret stored on itself in digital form. This improved resistance to physical attacks is the principal advantage of using a PUF.

The applications of Controlled PUFs are all the applications that can benefit from having a shared secret between a chip and a remote party. Digital rights management, set-top boxes and distributed computation are examples of such applications. More details can be found in [GCvDD02a].

4 Attacks

There are many possible attacks on silicon PUFs – we describe some of them in this section.

4.1 Duplication

To break the authentication methodology, the adversary can fabricate a "counterfeit" IC containing the PUF that produces exactly the same responses as the original IC/PUF for all challenges. A special case of this attack occurs when an IC manufacturer attempts to produce two identical ICs from scratch.

Given the statistical variation inherent in any manufacturing process, we argue that it is infeasible to produce an IC precisely enough to determine the PUF that it embodies. When producing two ICs in identical conditions (same production line, same position on wafer, etc.) the manufacturing variations are sufficient to make the two resulting PUFs significantly different. The probability that the two ICs will have identical PUFs 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 match. This is a very expensive proposition, both economically and computationally speaking.

We would like to draw the reader's attention to the fact that the process variations that we are building our security on cannot be easily eliminated by the manufacturer. These variations limit the manufacturer's ability to reduce IC feature size, and must also be taken into account when studying a circuit's timing constraints. Any reduction in process variation would directly lead to improved performance, so this is an active area of research. As an illustration, chapter 14 of [CK02] studies the impact of process variations on circuit design, and shows that as processes improve, relative variations increase rather than decrease.

It is because a silicon PUF is based on uncontrollable process variations, that we claim that silicon PUFs are manufacturer resistant (see section 1), at least in the case of ICs that are made in state of the art processes.

4.2 Timing-Accurate Model

Alternately, the adversary can attempt to create a timing-accurate model of the original PUF and simulate the model to respond to challenges, in effect creating a "virtual counterfeit." The accuracy of this model has to be comparable to the accuracy of reliable (on-chip) circuit delay measurement in order to produce a successful virtual counterfeit. Here, the adversary has three options, direct measurement, exhaustive enumeration of challenges, and model-building using observed responses based on a subset, i.e., a polynomial number of challenges.

4.2.1 Direct Measurement

The adversary can attempt to directly measure device delays of the circuit by probing or monitoring *internal* devices. He can then use these measured delays in a more or less sophisticated timing model.

In order to do this at the level of accuracy required to break authentication, he will have to remove the package and insert probes. Indeed, non-invasive attacks such as DPA [KJJ99] and EMA [QS01] extract information about collections of devices, not individual devices. Probing with sufficient precision is likely to be very difficult because the adversary runs the risk of changing the circuit delays while probing. Interactions between the probe and the circuit will directly influence the circuit. Moreover, in order to insert his probes, the adversary will potentially have to damage overlaid wires. Because of the high capacitive coupling between neighboring wires (see [DP97] for the importance of capacitive coupling between wires), damage to these overlaid wires could significantly change the delay that is to be measured.

How best to lay out the PUF circuit to make it highly sensitive to invasive attacks is a promising direction for further research.

4.2.2 Exhaustive Model

Clearly, a model can be built by exhaustively enumerating all possible challenges, but this is intractable, since there are an exponential number of possible challenges.

4.2.3 Model Building Using Challenge Subset

The adversary can use a publicly available mask description of the IC/PUF and apply challenges and monitor responses and attempt to build a timing-accurate model.

We first note that creating accurate timing models given mask information 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 determine polynomial-sized timing models that are accurate to within measurement errors, this would represent a breakthrough. However, the adversary has a slightly different problem – he needs to build a highly accurate model of a particular IC, to which he has access, and to which he can apply challenges and monitor responses.

The transient response of an IC is a non-linear and non-monotonic function of the delays of wires and devices in the IC. The adversary has to guess a general enough parameterizable model (e.g., delay of a device is dependent on load capacitance and transitions of neighboring devices), and obtain enough responses to well-chosen challenges such that he obtains a system of equations that can be inverted to obtain the parameters of his model.

We will discuss the barriers confronting the adversary in Section 5 for our chosen candidate PUF.

5 A Candidate PUF and Analysis of Model Building

Finding a delay circuit that produces a satisfactory PUF that is provably hard to break is difficult because of the numerous different types of attacks that are possible. It is unclear how classical hard problems such as factorization or discrete logarithm could be embedded in the analog behavior of a physical system. Therefore, we have to find our sources of hardness in other problems. This section shows our current best candidate PUF. Even if this circuit turns out to have vulnerabilities, it will certainly be possible to get around them. As our experience working with PUF circuits increases we expect to see their strength increase.

In order to analyze this candidate, we will assume an additive delay model. By this we mean that the delay of a path through the circuit is the sum of many individual wire and component delays. While this is a good approximation, it probably does not hold down to the precision of our measurements, so an adversary who is trying to build a model of the circuit would in fact have to break a system that is even more complex than the one presented here. In fact we believe that the sheer complexity of determining circuit delays precisely enough might even be sufficient to prevent modeling attacks on PUFs. Papers such as [LNPS00] show just how difficult precise delay simulation can be.

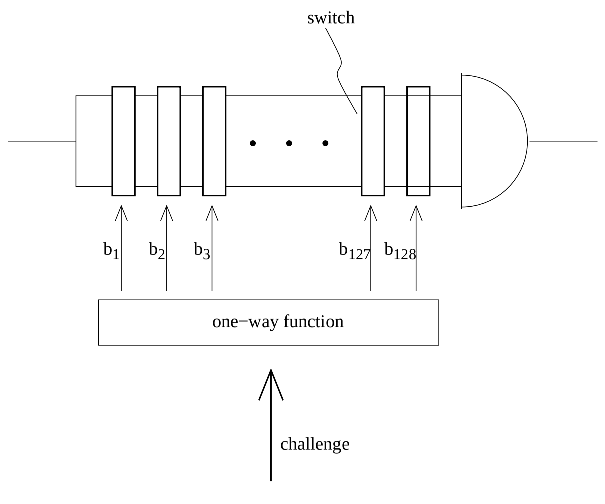


Figure 1: Circuit illustrating how paths are selected

The circuit for which we will measure delays that is implemented in our key card is depicted in Fig-

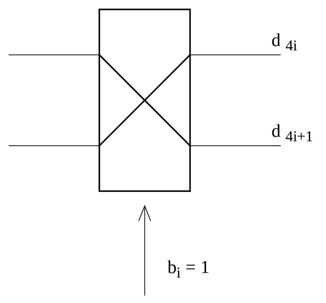


Figure 2: switch with $b_i = 1$

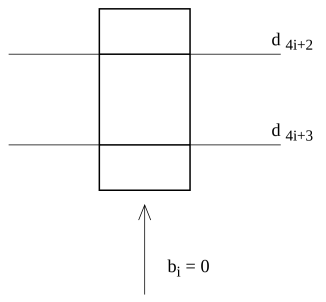


Figure 3: switch with $b_i = 0$

ure 1. A challenge of $n = 128$ bits is transformed by a one-way function into a bit pattern $\mathbf{b} = (b_1, \ldots, b_n)$. The bits b_i control switches. If $b_i = 1$, the switch is crossed (Figure 2); if $b_i = 0$, the switch is uncrossed (Figure 3). The input of the circuit is a wave with a single transition from 0 to 1. Initially, it is copied and the two waves are passed through the switches until they arrive at the AND gate. Depending on the delays from switch to switch one of the two wave fronts arrives first at the AND gate. The AND gate filters this wave front, only the second wave front is forwarded. The wave fronts follow complementary paths, the wave front following the path with the maximum delay is the output of the circuit. The total delay of a path is a sum of link delays and is determined in a linear way by the bits b_i. The response of the loop is in one to one correspondence with the maximum value of the two path delays.

The path maximizing the delay is called the *maximizing path of* $\mathbf{b}$. The maximizing delay can be derived from the response and is denoted by $m(\mathbf{b})$. Let $d_1, \ldots, d_{4n}$ be the different link delays of the circuit. We represent the two paths by sets $P_1(\mathbf{b})$ and $P_2(\mathbf{b})$ such that their corresponding delays are the sums $\sum_{i \in P_1(\mathbf{b})} d_i$ and $\sum_{i \in P_2(\mathbf{b})} d_i$ respectively. So either equation $m(\mathbf{b}) = \sum_{i \in P_1(\mathbf{b})} d_i$ or equation $m(\mathbf{b}) = \sum_{i \in P_2(\mathbf{b})} d_i$ holds. To the advantage of the adversary we assume no measurement noise and we assume that all link delays are constant and do not depend on the environment.

An adversary may input challenges of his choice and measure the corresponding responses. To argue his difficulty of building a model containing precise values for all the link delays d_i we show that

1. he has difficulty computing a linear set of equations solving all the link delays, and that

2. his resulting problem resembles the problem of sparsification of matrices for which the best known algorithm is exponential.

A challenge leads to 2 linear equations of which one is correct. Hence, two different challenges lead to 4 sets of 2 linear equations each, etc. In general k challenges lead to 2^k sets of k linear equations of which one set is the correct one. As each set of linear equations has $4n$ unknown delays, the adversary can use $4n$ challenges to do an exhaustive search among the 2^{4n} sets of linear equations to determine the correct set with which he can solve the link delays d_i and build a model. The 2^{4n} sets represent an exponential amount of work for the adversary using *poly*(n) challenges.

Without the one-way function an adversary can choose the bit pattern $\mathbf{b}$ without any restrictions and obtain the value $m(\mathbf{b})$. Let two patterns $\mathbf{b}$ and $\hat{\mathbf{b}}$ differ in two neighboring positions, for example $\mathbf{b} = (b_1, \ldots, b_i, 0, 0, b_{i+3}, \ldots, b_n)$ and $\hat{\mathbf{b}} = (b_1, \ldots, b_i, 1, 1, b_{i+3}, \ldots, b_n)$. Then, the maximizing paths of $\mathbf{b}$ and $\hat{\mathbf{b}}$ are most likely the same paths, only deviating in the $(i+1)^{th}$ and $(i+2)^{th}$ switch; in other words, if the top path of $\mathbf{b}$ is the maximizing path of $\mathbf{b}$, say, then the top path of $\hat{\mathbf{b}}$ is very likely to be the maximizing path of $\hat{\mathbf{b}}$. With this information,

$$A = \left(\begin{array}{cc|cc|c|cc}
\mathbf{b}^{0,1} & \mathbf{b}^{1,1} & \hat{\mathbf{b}}^{0,1} & \hat{\mathbf{b}}^{1,1} & \cdots & \tilde{\mathbf{b}}^{0,1} & \tilde{\mathbf{b}}^{1,1} \\
\mathbf{b}^{0,2} & \mathbf{b}^{1,2} & \hat{\mathbf{b}}^{0,2} & \hat{\mathbf{b}}^{1,2} & \cdots & \tilde{\mathbf{b}}^{0,2} & \tilde{\mathbf{b}}^{1,2} \\
\mathbf{b}^{0,3} & \mathbf{b}^{1,3} & \hat{\mathbf{b}}^{0,3} & \hat{\mathbf{b}}^{1,3} & \cdots & \tilde{\mathbf{b}}^{0,3} & \tilde{\mathbf{b}}^{1,3} \\
\vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
\mathbf{b}^{0,4n} & \mathbf{b}^{1,4n} & \hat{\mathbf{b}}^{0,4n} & \hat{\mathbf{b}}^{1,4n} & \cdots & \tilde{\mathbf{b}}^{0,4n} & \tilde{\mathbf{b}}^{1,4n} \\
\hline
m(\mathbf{b}) & m(\mathbf{b}) & m(\hat{\mathbf{b}}) & m(\hat{\mathbf{b}}) & \cdots & m(\tilde{\mathbf{b}}) & m(\tilde{\mathbf{b}})
\end{array}\right)$$

Figure 4: Matrix A

the adversary can halve the number of possible sets of equations, limiting his search space.

We can avoid this problem by using a one-way function as shown in Figure 1. However, accidentally two challenges may differ in a small number of coordinates after applying the one-way function. It can be shown that this probability is exponentially small in n.[4]

Fabrication process variations may lead to an asymmetric circuit in which one link delay is much larger than all the others. If this link delay is present in a path then this path will be the maximizing path and a linear equation is obtained. Such circuits can be easily modeled by an adversary. Our premise is that such large fabrication process variations occur with negligible probability. Further, if necessary, we can simply check to see if this is the case for each fabricated circuit and discard circuits with large variations because they can potentially be modeled.

The theoretical problem which the adversary needs to solve is a smart exhaustive search among all the possible sets of linear equations. Let us reformulate this problem. We represent the paths $P_j(\mathbf{b})$ by columns of 1's and 0's. The i^{th} coordinate is equal to 1 if and only if i is an element of the set $P_j(\mathbf{b})$. In this way we build a matrix with $4n$ rows, corresponding to the 4 delays for each switch, and $2c$ columns corresponding to 2 equations for each of c challenges. We add one extra row with the values $m(\mathbf{b})$. Let A be the resulting matrix. Figure 4 illustrates matrix A: $\mathbf{b}^{j,i}$ is the coordinate in the i^{th} position of the column corresponding to $P_j(\mathbf{b})$.

Let T be a vector which left multiplies matrix A and which consists of variables representing the $4n$ link delays and an additional entry -1. If T consisted of the actual link delays, then, if it is left multiplied with A, it creates a vector in which there is a zero in at least one of every two elements.

$$(d_1, d_2, d_3, \ldots, d_{4n}, -1) \cdot A = (*, 0, 0, *, 0, *, \ldots, *, 0)$$

illustrates the Gaussian elimination for obtaining zero entries in the last row of A. Thus, the goal of the adversary is to determine an instance of T which sparsifies the last row of A, that is, generates as many zero entries in the last row of A. The probability that this vector is not unique is exponentially small in $(c - n)$.

To sparsify matrix A an adversary may make use of side information about

1. the location of the zero entries[5], and
2. the constraints given by the max operation[6].

The circuit design leads to some structure in the matrices A as well. However, since the patterns $\mathbf{b}$ are selected by means of a random process (due to the one-way function) the matrices A have in this sense a random structure.

In general, without the side information, the best known algorithm for sparsifying any matrix has

[4] An error correcting code can be used to guarantee a large Hamming distance between patterns $\mathbf{b}$.

[5] Each pattern $\mathbf{b}$ gives rise to a zero entry in the last row of one of its corresponding columns.

[6] Both inequalities $m(\mathbf{b}) \geq \sum_{i \in P_1(\mathbf{b})} d_i$ and $m(\mathbf{b}) \geq \sum_{i \in P_2(\mathbf{b})} d_i$ hold.

a complexity exponential in the number of rows [EM98]. Taking the side information about the location of the zero entries into account, the complexity of the best known algorithm is still exponential in the number of rows. The adversary may be able to use the constraints, but there is no obvious way to significantly reduce the complexity by exploiting the constraints.

6 Experiments

In order to prove that identification is possible using delay variations between Integrated Circuits, we have implemented a PUF on Xilinx Spartan 2 FPGAs.[7] In these tests, identical circuits were placed on different FPGAs, and the resulting PUFs were compared. Our goal in this section is to show that the identification is possible given the measurement noise levels and manufacturing variations that we have observed.

6.1 Circuit Details

Because we do not have full control over the circuits that are implemented in an FPGA, a few compromises have to be made relative to the theoretical design.

First, the unpredictability of the circuit described in section 5 relies on having a circuit with a high level of symmetry between paths. The general purpose routing infrastructure of an FPGA makes it difficult to produce precisely matched paths. Therefore the FPGA circuits that we worked with do not have the degree of symmetry that would be required for a PUF to be secure. However, since the asymmetry is the same across all components, it does not make any change to the difficulty in identifying components, which is what we will be discussing in this section.

The second limitation of FPGAs, is that the lack of analog components makes it impractical to directly measure the delay of a path through the circuit with the precision that we require. To get around this problem, we use self-oscillating loops containing the path for which we want to measure the delay. Using digital circuitry, we can precisely measure the frequency of the self oscillating loops over a few tens of thousands of periods.

Note, however, that the use of self oscillating loops to measure delays is not ideal, and should not be used for a production design. First it drastically increases the time (and power) that is required to evaluate the PUF. Worse, it makes the frequency that is being measured, which is the response of the PUF to a challenge, vulnerable to differential power analysis. This is not very problematic for a key card application, but can be fatal in the case of Controlled PUFs (see [GCvDD02a]).

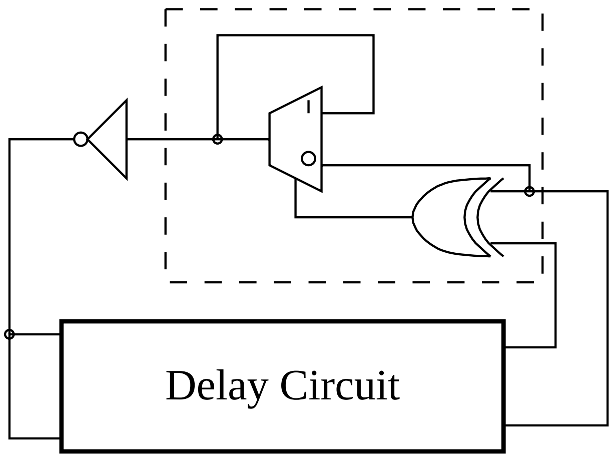


Figure 5: A self-oscillating circuit is built around the delay circuit. Measuring the frequency of the self-oscillating loop is equivalent to measuring the delay of a path through the delay circuit.

Figure 5 shows how a self oscillating loop is built around the delay circuit. Since this self-oscillating loop has to be used both for rising and falling transitions, the *and* gate that combines the two paths of the delay circuit of Figure 1 has been replaced by a more complicated circuit that switches when the slowest transition, be it rising or falling, reaches it. The circuit is essentially a flip-flop that changes state when both outputs from the delay circuit are at the same level.

The dotted box indicates a delicate part of the cir-

[7]The exact components that were used were the XC2S200PQ208-5.

cuit that cannot be implemented exactly as shown without running the risk of producing glitching in the output. In the FPGA it is implemented by a lookup table. In an implementation with simple logic, it should be implemented in normal disjunctive form. The representation that was made here was simply chosen for ease of understanding.

6.2 Robustness to Environmental Variation

So far, all our discussion has considered that path delays in a circuit are constant for a given component. In reality, this is far from the case. Environmental perturbations can account for variations that are large enough to mask out the small manufacturing variations that we are trying to measure. Therefore, they must be taken into account.

6.2.1 Temperature and Voltage Compensation

Parameters such as temperature or supply voltage can cause variations in delay that are orders of magnitude greater than the manufacturing variations that we are trying to observe. For a 30 degree Celsius change in temperature, the delays vary on the order of 5%. This is to be compared with inter-chip variations that are well below 1% on this size of circuit.

Fortunately, we have found that environmental variations operate roughly proportionally on all the delays in our circuit, and therefore, they can be compensated for by always working with delay ratios instead of absolute delays. Therefore, we place two different self-oscillating loops on the FPGA. We run both self-oscillating loops to get two frequencies, and take a ratio of the two frequencies as the PUF's response.

Once compensation has been applied, the variation with temperature is of the same order of magnitude as the measurement error.

Up to now, we have assumed that temperature is uniform across the integrated circuit. If that is not the case then temperature compensation is likely not to work well. With the circuit presented here, the paths are heated in a uniform way by the transitions that are running through them. With other circuits in which transitions only reach some parts of the circuit, we have observed non uniform heating which can cause unreliable measurement results. Therefore, we recommend the use of circuits that get heated in a uniform way during use.

6.2.2 Aging

Through prolonged use, the delays of an integrated circuit are known to shift. We have not yet studied the effect that aging might have on a PUF. In particular, if the changes due to aging are big enough, we might not be able to recognize a PUF after it has undergone much use. Studying these aging effects is an important aspect that must be covered by future work.

6.3 Identification Abilities

To test our ability to distinguish between FPGAs, we generated a number of profiles for many different FPGAs in different conditions. A profile is made up of 128 challenge-response pairs. All the profiles were established using the same challenges.

Two profiles can be compared in the following way: For each challenge look at the difference between the responses. You can then look at the distribution of these differences. If most of them are near zero, then the profiles are close. If they are far from zero then the profiles are distant. During our experiments, the distribution of differences was typically Gaussian, which allows us to characterize the difference between two profiles by a standard deviation.

Figure 6 shows the differences between the profile for an FPGA called Abe on Blaise's test board at room temperature, and a number of other profiles (σ is the standard deviation):

- Another profile of Abe on Blaise's test board at room temperature ($\sigma \approx 1 \cdot 10^{-5}$). (This reflects power supply variations with time at a reader.)
- A profile of Abe on Tara's test board at room temperature ($\sigma \approx 2.5 \cdot 10^{-5}$). (This reflects power supply variations across card readers.)

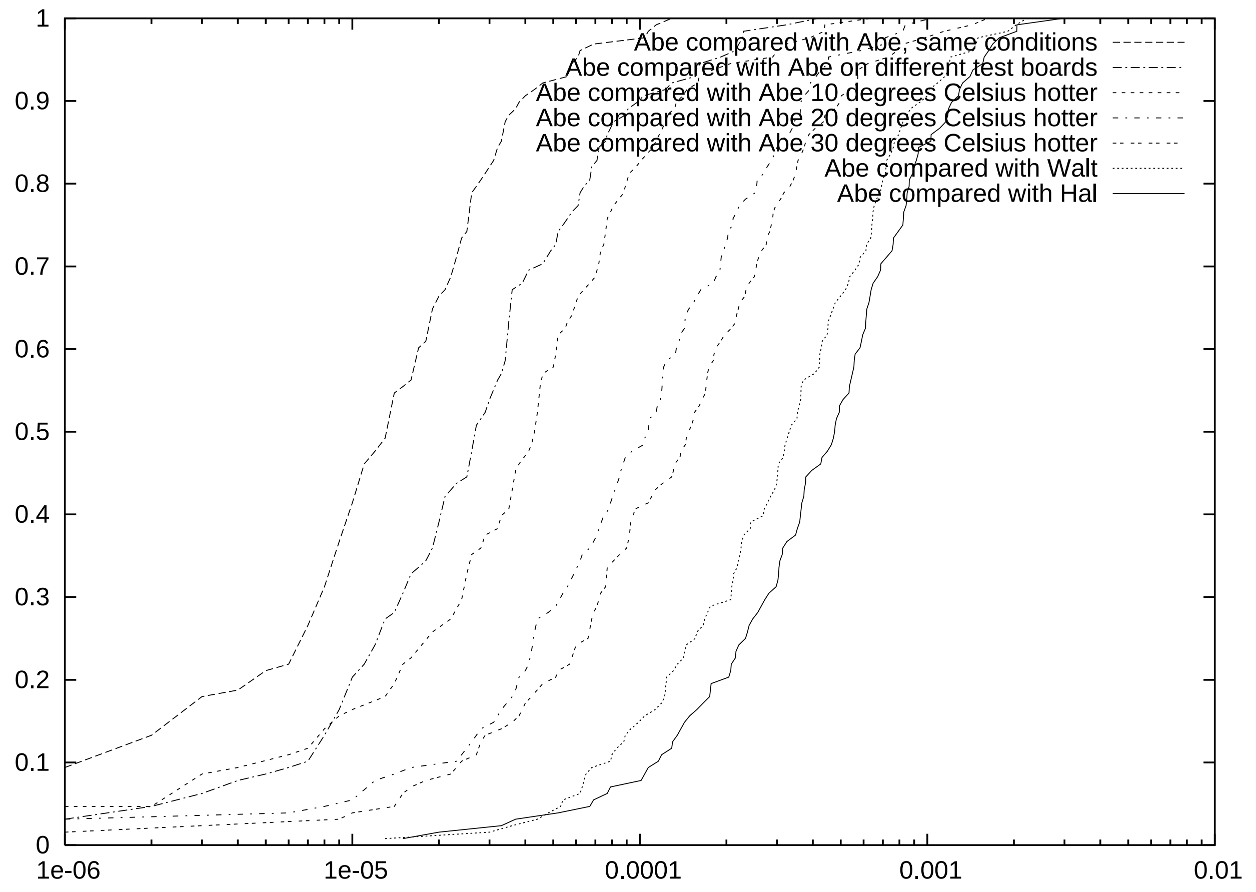


Figure 6: Comparing the FPGA called Abe at room temperature with itself in various conditions, or with other FPGAs. The vertical axis indicates the probability that for a given challenge, the difference in response will be lower than the difference in response that is indicated on the horizontal axis. These plots illustrate the typical behavior we encountered in our experiments with many FPGAs.

- Profiles of Abe on Blaise's test board at 10, 20 and 30 degrees Celsius above room temperature ($\sigma \approx 5 \cdot 10^{-5}$ to $1.5 \cdot 10^{-4}$).
- Profiles of FPGAs Hal and Walt on Blaise's test board at room temperature ($\sigma \approx 4 \cdot 10^{-4}$).

The above standard deviations were typical across different FPGAs and comparisons of different pairs of FPGAs.

Clearly, it is possible to tell FPGAs apart. Though our ability to tell them apart depends on how much environmental variation we need to be robust to. Even with 30 degree Celsius variations, each challenge is capable of providing 0.7 bits of information about the identity of the FPGA. This goes up to 1.5 bits if only 10 degree Celsius variations are allowed.

If we want to distinguish between 1 billion different components we need a sufficient number of bits to identify $10^{18} \approx 2^{60}$ components (this is because of the birthday phenomenon). Getting those 60 bits of information requires from 40 to 90 challenges depending on the temperature variations that we are willing to tolerate.

The numbers that are given here are very dependent on the PUF circuit that is considered. In the circuit that we studied in [GCvDD02b] we had a signal to noise ratio that was much better than we observed in the current circuit. We believe that by paying more attention to how our circuit is laid out, we will be able to build PUFs for which more bits can be extracted from each challenge.

7 Conclusion

We have presented a technique for delay-based circuit authentication and conducted preliminary experiments that show that it is viable. By using this method, it is possible to store secrets on a chip in a way that is less vulnerable to invasive attacks than traditional digital methods.

More experiments are necessary to gauge the reliability of authentication under environmental variations, including circuit aging.

We argued that delay-based authentication is not susceptible to conventional attacks that attempt to discover a secret, hidden key. One of the most plausible attacks is model building. The particular circuit we experimented with is a simple, symmetric circuit, for which it appears that model building is quite hard, though we are still lacking a formal proof.

While a number of problems need to be solved in order to use delay-based authentication in applications such as smart card authentication and software licensing, we believe that this is a promising direction for future research.

Acknowledgements

We thank Tara Sainath and Ajay Sudan for help with the FPGA experiments.

References

[AK96] 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.

[AK98] 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.

[And01] Ross J. Anderson. *Security Engineering: A Guide to Building Dependable Distributed Systems.* John Wiley and Sons, 2001.

[Ber98] Kerry Bernstein. *High Speed CMOS Design Styles.* Kluwer Academic Publishers, 1998.

[BN00] D. S. Boning and S. Nassif. Models of Process Variations in Device and Interconnect. In W. Bowhill A. Chandrakasan and F. Fox, editors, *Design of High Performance Microprocessor Circuits*, chapter 6. IEEE Press, 2000.

[CK02] David Chinnery and Kurt Keutzer. *Closing the Gap Between ASIC & Custom.* Kulwer Academic Publishers, 2002.

[DP97] Florentin Dartu and Lawrence T. Pileggi. Calculating worst-case gate delays due to dominant capacitance coupling. In *Proceedings of the 34th annual conference on Design automation conference*, pages 46–51. ACM Press, 1997.

[EM98] Sebastian Egner and Torsten Minkwitz. Sparsification of rectangular matrices. *Journal of Symbolic Computation*, 26(2):135–149, 1998.

[GCvDD02a] Blaise Gassend, Dwaine Clarke, Marten van Dijk, and Srinivas Devadas. Protocols and applications for controlled physical unknown functions. In *Laboratory for Computer Science Technical Report 845*, June 2002.

[GCvDD02b] Blaise Gassend, Dwaine Clarke, Marten van Dijk, and Srinivas Devadas. Silicon physical unknown functions. Technical report, MIT LCS TR-833, May 2002.

[KJJ99] P. Kocher, J. Jaffe, and B. Jun. Differential Power Analysis. In *Proceedings of Crypto'99*, 1999.

[LDT00] K. Lofstrom, W. R. Daasch, and D. Taylor. IC Identification Circuit Using Device Mismatch. In *Proceedings of ISSCC 2000*, pages 372–373, February 2000.

[LNPS00] Ying Liu, Sani R. Nassif, Lawrence T. Pileggi, and Andrzej J. Strojwas. Impact of interconnect variations on clock skew of a gigahertz microprocessor. In *Design Automation Conference.* IEEE/ACM, June 2000.

[MvOV96] Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography.* CRC Press, 1996.

[QS01] Jean-Jacques Quisquater and David Samyde. ElectroMagnetic Analysis (EMA): Measures and Counter-Measures for Smart Cards. In *Proceedings of Smart Card Programming and Security (E-smart 2001), LNCS 2140*, pages 200–210, September 2001.

[Rav01] P. S. Ravikanth. *Physical One-Way Functions.* PhD thesis, Massachusetts Institute of Technology, 2001.

[SW99] 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.

[WE85] N. Weste and K. Eshraghian. *Principles of CMOS VLSI Design: A Systems Perspective.* Addison Wesley, 1985.