CONCURRENCY AND COMPUTATION: PRACTICE AND EXPERIENCE
Concurrency Computat.: Pract. Exper. 2003; **3**:1–20 *Prepared using* cpeauth.cls *[Version: 2002/09/19 v2.02]*

Identification and Authentication of Integrated Circuits

Blaise Gassend[1], Daihyun Lim[1], Dwaine Clarke[1],
Marten van Dijk[2], Srinivas Devadas[1]*

[1] *Massachusetts Institute of Technology, Laboratory for Computer Science, Cambridge, MA 02139, USA*
[2] *Prof Holstlaan 4, Eindhoven, The Netherlands*

SUMMARY

This paper describes a technique to reliably and securely identify individual integrated circuits (ICs) based on the precise measurement of circuit delays and a simple challenge-response protocol. This technique could be used to produce key-cards that are more difficult to clone than ones involving digital keys on the IC. We consider potential venues of attack against our system, and present candidate implementations. Experiments on Field Programmable Gate Arrays show that the technique is viable, but that our current implementations could require some strengthening before it can be considered as secure.

KEY WORDS: Authentication, identification, physical random function, physical security, smartcard, tamper resistance, unclonability

1. 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 [13]. 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 attempts by the adversary to determine the key destroys the key. 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 physical attacks is a challenging problem and is receiving some attention [3]. 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 [11], 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 [17]. 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 [1, 2].

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 [11] and electromagnetic analysis [15] are not as well suited for breaking delay-based authentication. Another 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. 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: We describe the notion of a physical random function, which is what we are trying to implement in Section 2. An overview of our approach to identify and authenticate ICs based on delays is given in Section 3. Then, in Section 4 we describe some applications, in particular a secure key card application. We consider plausible attacks on PUFs in Section 5. Section 6 presents a candidate PUF circuit based on delay measurement, and discusses its identification characteristics. Section 7 presents a second candidate circuit that compares two delays, and goes over relevant experimental measurements. Finally, Section 8 shows how these candidate circuits can be attacked in the additive delay model.

2. Definitions

Definition 1. *A Physical Random Function (PUF)*[†] *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 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 [16] 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 [14]. A PUF *is* a one-way function in the sense that it is hard to reconstruct the physical system from challenge-response pairs. However, unlike a one-way function, a PUF does not require going from the response to the challenge to be hard. For a PUF, all that matters is that going from a challenge to a response without using the device is hard.

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 systems that are unique without having to trust the manufacturer of the device.

In this paper, the PUFs we produce from delay measurements don't truly fit the *hard to predict* criterion of the PUF definition, as a part of the response may be easy to predict. There are methods to get around this subtle difference that are not described here [7, 8].

3. Delay-Based Authentication

3.1. 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;

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

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 gates (cf. Chapter 12 of [5]). Delay variations of the same wire or device in different dies have been modeled using Gaussian distributions and other probabilistic distributions [4].

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 parts per million.

3.2. 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. However, temperature and power supply voltage have a significant effect on the absolute values of circuit delays [19]. To keep the signal-to-noise ratio high under significant environmental variations, we require compensated delay measurement (cf. Section 6.2.1). Using compensated delay measurement, we can keep the variation due to standard deviation sufficiently below inter-chip variation to allow reliable identification despite a wide range of environmental variations.

3.3. Generating Challenge-Response Pairs

As we mentioned in the introduction, manufacturing variations have been exploited to identify individual ICs. However, the identification circuits used so 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 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 $\frac{1}{2}log_2(1+\sigma/\delta)$ (though this limit is difficult to reach in practice). By using multiple independent challenges, we can extract a large number of identification bits from an IC. Actually producing a huge 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 the 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.

4. Applications

4.1. Secure Keycard

The simplest application for PUFs is to make tamper-resistant, unforgeable key cards. This application was first described in [16]. We will argue in Section 5 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.

4.2. 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 [16], 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 [7, 9], 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 [7, 9].

5. Attacks

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

5.1. 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 [5] 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 (cf. Section 2), at least in the case of ICs that are made in state of the art processes.

5.2. 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.

5.2.1. Direct Measurement

The adversary can attempt to directly measure device delays of the circuit by probing or monitoring components inside the device. 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 differential power analysis [11] and electromagnetic analysis [15] 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 (cf. [6] 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 direction for further research.

5.2.2. 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.

5.2.3. 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. The PUF designer's job is to make this task as difficult as possible.

In Section 8 we will return to modeling attacks when considering the difficulty of modeling our proposed authentication circuits.

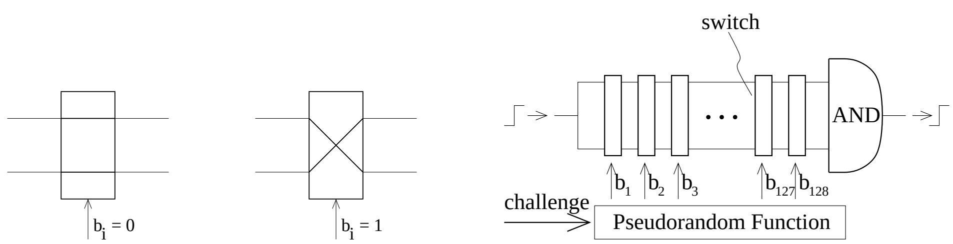


Figure 1. The switch block defines different circuits for challenge bits of zero (a) or one (b). In the MAX-delay circuit, they are combined into a long chain (c).

6. The MAX Circuit

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.

This section shows a candidate circuit that we have performed experiments on. We conjectured in [10] that this circuit would be difficult to break in the additive delay model, in which we assume that the delay of a complex circuit can be exactly broken up into a sum of delays of components that make it up. More recent work that is presented in Section 8 suggests that our conjecture was incorrect. Therefore, to create a PUF that is secure in the additive delay model, modifications will be needed to make this circuit more difficult to analyze. Section 8.4 suggests some improvements that could be made. We believe that the sheer complexity of determining circuit delays precisely enough as the circuit gets more complicated will turn out to be sufficient to prevent modeling attacks on PUFs. Papers such as [12] show just how difficult precise delay simulation can be.

The circuit for which we will measure delays is depicted in Figure 1(c). A challenge of $n = 128$ bits is transformed by a pseudorandom function into a bit pattern $\mathbf{b} = (b_1, \ldots, b_n)$. The bits b_i control switches. If $b_i = 0$, the switch is uncrossed as in Figure 1(a). If $b_i = 1$, the switch is crossed as in Figure 1(b). To get a response from this circuit, we present a rising edge on its input. That edge is split into two competing edges that independently propagate through the switches until they reach the AND gate. When the slowest of the two edges reaches the AND gate, an edge appears on the output of the circuit. The response of the circuit is the time it took for the edge on the input to produce an edge on the output.

In the additive delay model, this response of the circuit for a given challenge can be expressed as the max of two sums of elementary path delays. Thus we clearly see the need for the AND gate. Without it, responses would be linear combinations of elementary delays, and obtaining elementary delays from challenge-response pairs would reduce to solving a linear system of equations. With the AND gate, each challenge-response pair produces two candidate equations, only one of which actually holds. Ideally for k pairs, the adversary would have 2^k different systems of equations to choose from, though we expect an intelligent adversary to be able to eliminate many of these possibilities.

If the adversary is able to directly chose the b_i that gets presented to the device then there are easy ways for him reduce the number of systems of equations he must deal with. For example, by choosing two sets of b_i that only differ for two neighboring stages, there is a high likelihood that the same path was the slowest in both cases, in this way only two of the four possible systems of equations need to be considered. With a little more work, the individual delays in the circuit can once again be found by solving a linear system of equations. To prevent this type of attack, we do not allow the adversary to directly chose the b_i values directly. An error correcting code would be sufficient to prevent the attack we described by forcing the adversary to chose different values of b_i that have a minimum Hamming distance from each other. We prefer to use a pseudorandom function as it allows us to prevent any kind of *chosen challenge attack*, even though the minimum Hamming distance property isn't guaranteed.

One more precaution is necessary. If one elementary delay is much longer than all the others, then it will always be in the slowest path through the circuit, which once again reduces the adversary's work to solving a linear system of equations. This problem is particularly serious if the same elementary delay is particularly long on all instances of the device. To prevent this problem, the delay circuit we request that the circuit be designed to be as symmetrical as possible. That way, knowledge of the circuit layout cannot help the adversary guess which path through the circuit is the slowest. Moreover, with these precautions, the likelihood that one elementary delay will be much longer than the others through random process variations is kept to a minimum.

6.1. Circuit details

In order to prove that identification is possible using delay variations between Integrated Circuits, we have implemented a PUF on Xilinx Spartan 2 FPGAs.‡ 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.

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 MAX-circuit 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

‡The exact components that were used were the XC2S200PQ208-5. We would like to thank Tara Sainath and Ajay Sudan for help with these experiments.

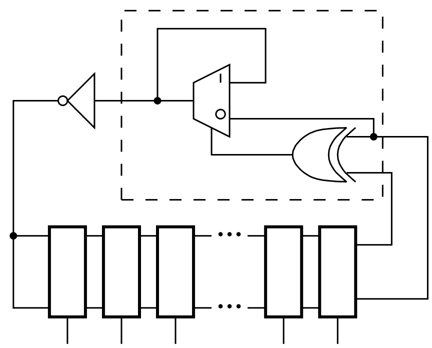


Figure 2. Measuring the MAX circuit with a self-oscillating loop

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 [9]. Section 7 shows an alternative measurement method.

Figure 2 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(c) 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 circuit 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 used 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 true. Environmental perturbations can account for variations that are large enough to mask out the small manufacturing variations 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 cause variations in delay that are orders of magnitude greater than the manufacturing variations we are interested in. 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 only slightly above measurement error. However, as the variation increases, so does the error. For very large temperature changes (30 degrees Celsius is getting close to the limits), we can no longer expect to reliably recognize a PUF. If more temperature variation needs to be accommodated, it is possible to characterize the PUF once when it is hot and once when it is cold (more steps are possible for large temperature ranges). During use, one of these two cases will apply, so the PUF will be correctly recognized.

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. Interference With Other Sub-Systems

Another kind of environmental interference that has to be considered is the interaction between a self-oscillating loop, and other circuitry on the integrated circuit.

Experiments in which we measure the frequency of a loop oscillating alone, or at the same time as other loops show that the interference is very small. This has been demonstrated in [8] where the interference was provided by seven self-oscillating loops, and once again in our latest experiments where the frequencies of the two loops that are being measured can be measured simultaneously or successively. In each case, the interference caused by the other self-oscillating loops is of the same order of magnitude as measurement error.

There is however one case in which interference is non negligible. It is the case when the interference is at almost the same frequency as the self-oscillating loop. In that case, the loop's

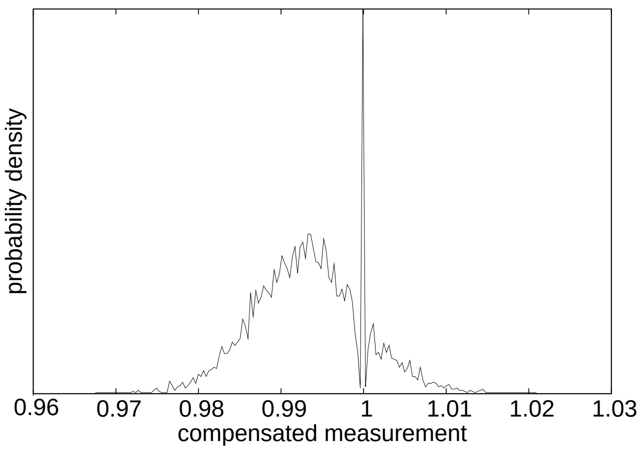


Figure 3. Distribution of responses to randomly selected challenges. Each response is the ratio of the frequencies of two simultaneously-running loops. As can be seen, when the loop frequencies are too close, the loops lock and the response is unity.

frequency tends to lock on the perturbating frequency. Because of this, we recommended not to simultaneously measure the two frequencies that will get combined into a compensated measurement. Figure 3 shows the result of locking on compensated measurements: values near unity have been forced towards unity by the locking phenomenon.

6.2.3. 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.

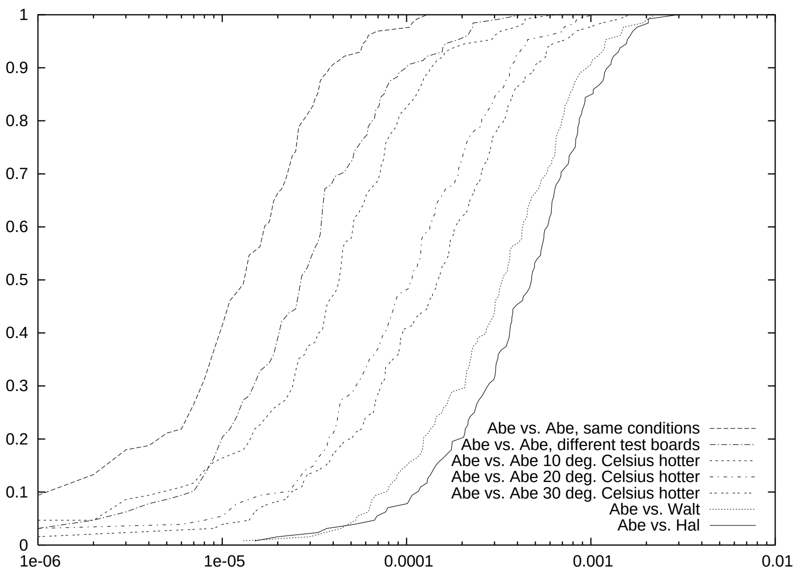


Figure 4. Comparing the FPGA called Abe at room temperature with itself in various conditions, or with other FPGAs. The vertical axis indicates the probability that for a given challenge, the difference in response will be lower than the value in parts per million that is indicated on the horizontal axis. These plots illustrate the typical behavior we encountered in our experiments with many FPGAs.

Figure 4 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 characterizes measurement noise.
- A profile of Abe on Tara's test board at room temperature ($\sigma \approx 2.5 \cdot 10^{-5}$). This gives an idea of the effects of power supply variations between card readers.
- 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 able to tolerate. 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 paradox). Getting those 60 bits of information requires from 40 to 90 challenges depending on the temperature

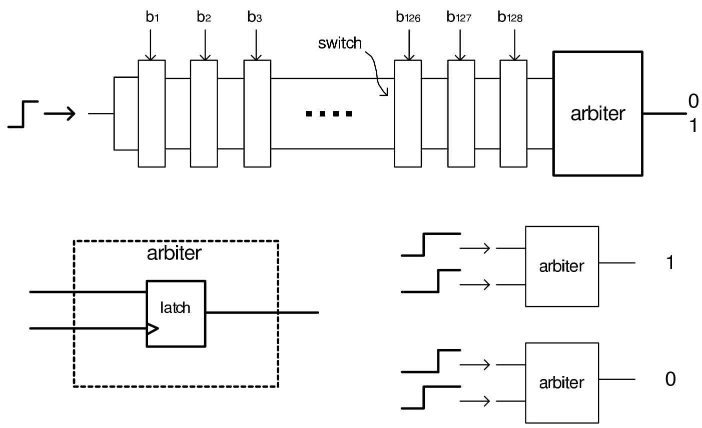


Figure 5. The arbiter circuit is like the MAX circuit except that the AND gate is replaced by an arbiter. The arbiter decides which of the two signals reaches it first and outputs a bit.

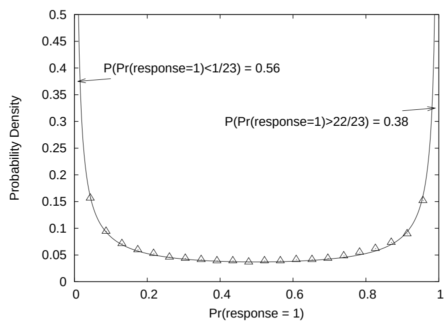
variations that we are willing to tolerate. We assume that the responses are essentially independent of each other because, if we assume that each stage contributes a few bit of information, then we find that there is an order of magnitude more information in the circuit than we need.

The numbers that are given here are very dependent on the PUF circuit that is considered. In the circuit that we studied in [8] 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.

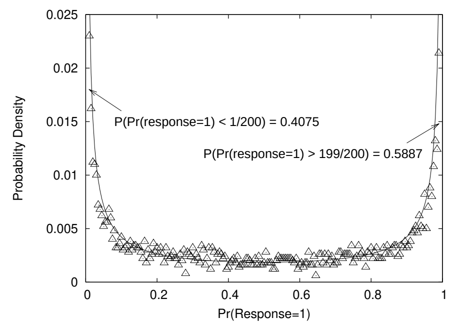
7. The Arbiter Circuit

Measuring delays using self-oscillating circuits as we did in Section 6 is easy and precise. However, to get a precision in the hundreds of parts per million requires tens to hundreds of thousands of clock cycles, which implies that thousands of edges will have to be propagated through the delay circuit each time we want to measure a single delay. This means that measurement is slow. To make delay measurement faster, we would like to do a more direct measurement, which only needs to run a single edge through the delay circuit. This is not a trivial task given that in our experiments, we have to measure the delays with precisions on the order of tens of picoseconds.

The method that we have chosen to explore is to use an *arbiter*. The arbiter has two inputs, which are both low initially. The arbiter waits for one of the inputs to go high, at which time its output indicates which input went high first.



(a) Inter-Chip variation



(b) Repeated measurement

Figure 6. Probability distribution for a challenge of outputting 1: across multiple different chips (a), or when measurement is made repeatedly on the same chip (b). Triangles show experimental data, while the curve shows a theoretical fit. For the inter-chip variation, an ideal implementation would have an inverted curve which the highest likelihood near 50%.

Figure 5 shows the structure of arbiter circuit and its operation. We are assuming as for the MAX circuit that the challenge has been passed through a pseudorandom function before being fed into the delay circuit. Two signals race through the delay paths that are defined by the challenge input bits. At the end of circuit, the arbiter decides which signal arrived first and outputs a bit.

If the difference in delay for the two racing paths isn't much larger than the inter-chip variation in delay, then the value of the output bit is likely to be different from one chip to another, which makes identification possible. For this reason, it is highly desirable for the arbiter circuit to be designed to be as symmetrical as possible. This way, all the challenges will have outputs that vary from chip to chip. Unfortunately, as we shall see, our FPGA implementation was not very symmetrical so most of the bits have little identification value for us, which increases the number of measurements that have to be made before a device can be identified.

7.1. Experimental Results

7.1.1. Inter-Chip Variation

To study the characteristics of our FPGA implementation of the arbiter circuit, we measured responses for 100,000 challenges across 23 FPGAs, on 16 different layouts of the delay circuit. For each challenge-layout pair, we calculated the probability of a response bit being 1.

Figure 6(a) shows the distribution of those probabilities. Assuming that delays have a Gaussian variation from chips to chip, and a Gaussian variation from challenge to challenge, the measurements we made suggest that there is about 25 times more variation due to challenges than to inter-chip variation. Only 6% of challenges actually have a response that changed at least once from chip to chip in our experiments. We are currently producing an ASIC test chip with a delay circuit that was laid out by hand to maximize symmetry. This way most of the challenges should produce useful identification information (or none at all if a skew is introduced between the two paths).

To study how different chips are from each other, we look at the number of challenges in our 100,000 challenge test set that change from one chip to another; this is called the Hamming distance. In our experiments with 23 FPGAs, the average Hamming distance between chips was 1049, roughly 1.05% of the challenges.

7.1.2. Measurement Noise

To evaluate measurement noise, we measured a set of 100,000 challenges 200 times each, on 16 different layouts of the delay circuit. Challenges that didn't return the same value each time were counted as unreliable. In this way, we found that about 0.1% of challenges were unreliable on the PUF arbiter circuit. This is more than ten times less than the distance between two distinct components; identifying chips using this circuit is easy.

When considering this number, one should keep in mind that the challenges that are unreliable are most likely the ones that have comparable delays for both of the racing paths, i.e., precisely the ones that are useful for identification purposes. Therefore, in a more symmetrical circuit, the number of useful challenges for identification will increase hand in hand with the measurement noise.

Figure 6(b) shows how the probability of outputting 1 is distributed for the set of challenges and layouts that was measured. The same Gaussian based analysis we did for inter-chip variation shows that there is about 450 times more variation from challenge to challenge than due to measurement noise, that is about 18 times more inter-chip variation than noise.

7.1.3. Temperature variation

A nice aspect of the arbiter circuit is that it compares the delays of two paths through the circuit. Thus, it automatically does temperature compensation. This makes post processing simpler than it was for the MAX circuit.

Figure 7 shows the amount of noise introduced by temperature variation. We made a profile for 100,000 challenges at the reference temperature of 28 degrees Celsius and calculated its distance from the response vector at increasingly higher temperatures. Even with a change in temperature greater than 40 degrees Celsius, the noise is only about 0.3%, well below the inter-chip variation of 1.05%

Overall, the arbiter circuit has qualitatively comparable performance to the delay measurement circuit. If the proportion of challenges that are useful for identification can be increased, it will be much faster to use than the delay measurement circuit, which makes it an excellent candidate for PUF applications.

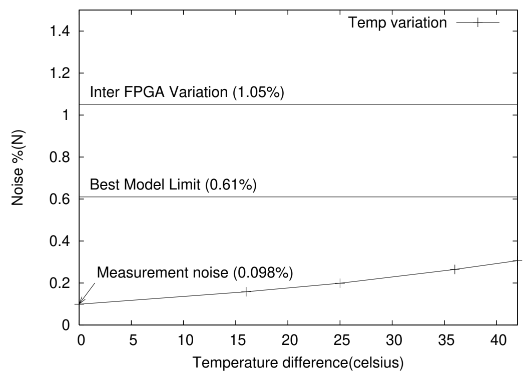


Figure 7. Variation in normalized Hamming distance when a chip is compared with itself at a different temperature. The inter-chip variation and our best model-building attack are shown for comparison.

8. Modeling Attacks

As we saw in Section 5.2, it must be difficult for attackers to build an accurate timing model of our circuit if we want to prevent them from impersonate it. To evaluate the difficulty of this type of attack we have tried to carry it out ourselves. The results we have to date suggest that the basic MAX and arbiter circuits are not difficult enough to model, contrarily to what we had conjectured in [10].

8.1. Additive-Delay Model

We model the circuits using an additive delay model. In this model, we assume that the delay of a path through the circuit is the sum of the delays of the elementary components that make it up. Knowing all the elementary component delays an attacker can calculate the circuit delay for any challenge. The attacker's task is to find the elementary delays. Assuming that getting the elementary delays by opening the circuit and doing direct measurement is impractical, we see that an attacker will have to infer those delays from the outputs of the device on a number of challenges. Because of the pseudorandom function that has been placed in the circuit, the challenges that the adversary has to work with are effectively random.

The circuits we have considered in sections 6 and 7 are made up of a sequence of identical stages, each containing a switch block. The switch block can be modeled by using 4 delays as shown in Figure 8(a). But this representation contains redundancy, and does not reveal all the structure of the system of delays. Indeed, the circuit can be rewritten as in Figure 8(b). In this figure, the delays have been separated into four distinct components: $u = (a+b+c+d)/4$ that is common to all paths, $v = (a+b-c-d)/4$ for the difference between crossed and straight paths, $x = (a-b+c-d)/4$ for the difference between paths that start at the top and paths

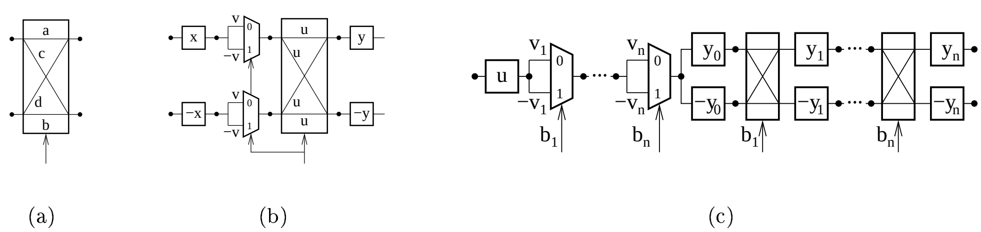


Figure 8. Elementary switch model before (a) and after (b) simplification. The resulting simplified model of a switch-based delay circuit (c). Delays are measured between dots.

that start at the bottom, and $y = (a - b - c + d)/4$ for the difference between paths that end at the top and paths that end at the bottom. When we put all the stages together, the rewritten form is easy to simplify. Except at the first stage, the x terms merge with the y terms from the previous stage. Because they are identical for both paths and commute with the switches, the u and v terms can be moved to the front of the circuit before the paths branch apart. Finally all the u terms can be merged. What remains is depicted in Figure 8(c).

All together the n switched stages are modeled by using $2n + 2$ parameters instead of $4n$. This result was to be expected as there are $2n - 2$ wires that connect two stages; where exactly along each wire we choose to end one switch and start the next is irrelevant which leads a redundant parameter. For the arbiter circuit, only $n+1$ parameters are necessary, as the terms that appear before the paths split are common to both paths, and only the difference between the two delays is considered by the arbiter.

Algebraically, the delay of one path can be expressed as

$$u + \sum_{i=1}^{n} (-1)^{b_i} v_i + \sum_{i=0}^{n} (-1)^{p_i} y_i$$

where the parity vector $p_i = b_i \oplus \cdots \oplus b_n$ is the number of times the paths exchange between the end of stage i and the end of the circuit, and $\oplus$ denotes the XOR operation. The delay of the other path is similar except that the y_i terms change sign. The delay can be seen as the sum of a constant u, the dot-product of the challenge vector b_i with a vector v_i, and the dot-product of the parity vector p_i with a vector y_i. For the arbiter case only the second dot-product remains as the rest is common to both paths.

8.2. The Perceptron Algorithm

The perceptron algorithm [18] is the simplest available algorithm for training single layered neural networks. In such a network, the output of a neuron is computed by taking the dot-product of an input vector (typically a vector of +1 and -1) with a vector of weights, and comparing the result with a threshold. To train a neuron on a set of input vectors, one simply iterates through the input vectors. If an input produces the correct output, nothing is changed. If an input produces an incorrect output, each one of the neuron's weights in changed in the direction that would tend to make it produce the correct result.

If the training set can be learned by the neural network (i.e., if the two output values can be separated by a hyper-plane in the input space), then the perceptron algorithm converges to a solution that satisfies all the training cases. If not, a slightly improved version of the algorithm can be used, which converges to the set of weights that maximizes the number of training cases that the neuron correctly decides.

According to the analysis we did in Section 8.1 the arbiter circuit can be viewed as a neuron, where the weights are a function of the elementary circuit delays, and the input vector is a simple function of the challenge bits. Therefore, we can expect the arbiter circuit to be very easy to build a model for under the additive delay model. What remains to be seen is whether the additive delay model is precise enough to produce a model that can be used to impersonate a physical device.

Because of the MAX operation, and because it has a real valued output, the MAX circuit cannot be reduced exactly to the perceptron algorithm, however, we found that a slightly modified algorithm gives excellent results. The method is as follows: first the model is initialized with random elementary delays. Then for each example in the training set, the output of the circuit is calculated. Based on how far the output is from the desired output, the elementary delays on the path that was selected by the MAX function are adjusted to bring the output closer to the desired value. In numerical experiments, we found that this method usually perfectly learned the training set (in rare cases it gets stuck far from the correct solution, but slightly different starting parameters then lead to perfect convergence). When noise was added to the experiments, the algorithm converged to within the amount of noise that was added.

At this point, we must conclude that if the additive delay model applies sufficiently precisely to our circuits, then a model building attack is possible and even easy. This does not necessarily mean that the real circuits can be modeled though, as the additive delay model is not precise enough to properly model the real device.

8.3. Modeling Experiments

To try to get an idea of how much our circuit deviates from the additive delay model, we attempted to train a model of the arbiter circuit using 90,000 challenge-response pairs measured on our FPGA implementation. When evaluated on a different set of 100,000 challenge-response pairs, the failure rate of the model was only about 3.3%. This is greater than the inter-chip variation, so one would be better off using a different FPGA than using our model when attempting to impersonate a chip.

Our initial hypothesis for these disappointing (from the attacker's point of view) results was that we were getting interference between the two paths through the delay circuit. Indeed, the switch component in the delay path is implemented by using a pair of multiplexers, that each have the challenge bit and both of delay paths as inputs. In Xilinx FPGAs, combinational logic is mainly implemented using 4 input lookup tables. In particular, in the circuit we implemented, the multiplexers were being implemented as lookup tables. These lookup tables are laid out

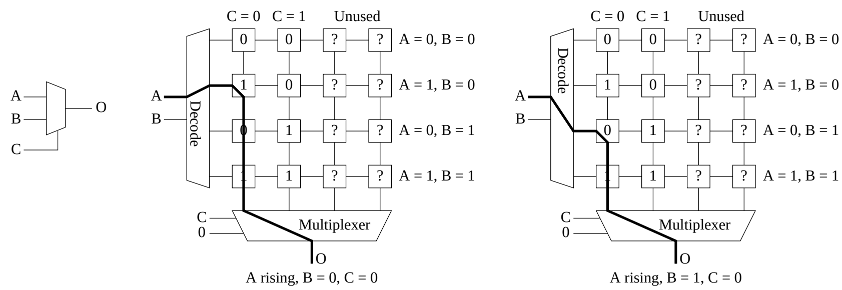


Figure 9. In a lookup table implementation of a multiplexer, the path that is taken by a rising edge on input A to get to output O depends on the state of input B.

as 4 by 4 grids of SRAM cells as shown in Figure 9. As the figure shows, a rising edge that goes through the multiplexer takes a different path through the grid depending on the value of the spectator input that is not currently selected. To take this difference into account would require a more complex model than the one we had considered.

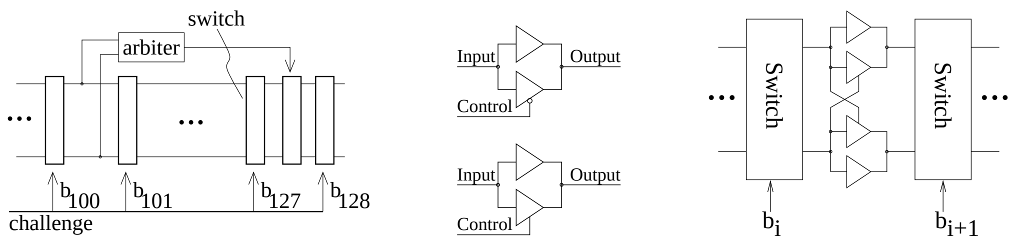
To verify this hypothesis, we re-synthesized our circuit using the MUXCY component that is part of the FPGA's fast carry logic, hoping that this component would be implemented in such a way that the path through the multiplexer does not depend on the spectator input. This was a reasonable assumption as most multiplexer implementations have this property. This time, the perceptron algorithm gave us a model that had a failure rate of 0.61%. This is still above measurement noise, even with 40 degree Celsius variations in temperature, but is well below inter-chip variation (cf. Figure 7).

At this point, we cannot say that the arbiter circuit of Figure 5 is broken, but an extremely simple modeling effort has been able to get uncomfortably close to measurement noise for one implementation of the arbiter circuit. Similar experiments are currently underway for the MAX circuit.

8.4. Hardening the Circuit

Our experimental results suggest that the circuits we have proposed to date are not as strong against modeling attacks as we would like. A number of directions can be taken to try to make the circuit stronger, evaluating these options will be the object of future work.

Feed forward arbiter: An arbiter is placed at an intermediate point in the circuit. Its output drives one of switches later on in the circuit as in Figure 10(a). Many such arbiters could be added to the circuit for increased difficulty. The major problem with this approach is that it adds discontinuities in the circuit, i.e., conditions in which a slight change in an elementary delay (due to noise, for example) can lead to a huge difference in the delay of a path. Moreover, this circuit can easily be expressed in the additive delay model, and we fear that a variant of the perceptron algorithm would be able to break it.



(a) (b) (c)

Figure 10. Candidate methods to defeat additive delay modeling: feed-forward arbiter to generate unknown challenge bits (a), variable-delay buffer implementation (b) and use (c).

Variable delay buffers: Here we try to add interactions between the paths through the circuit, reproducing the effects of the lookup-table multiplexer implementation. This is done by using what we call a variable delay buffer (cf. Figure 10), a component that directly passes a signal from its input to its output, but that does so at a different speed depending on the state of its other input. This approach is nice as it leads to continuous but non-monotonic circuits (the delay of the circuit is a continuous function of the elementary circuit delays, but the total delay might decrease as the delay of an elementary delay increases). Moreover, by putting some effort into the delay characteristics of the variable delay buffer, the circuit can be made hard to model in the additive delay model.

Change the circuit topology: So far we have only explored an extremely simple circuit topology, there are scores of other possibilities left to explore. The three things to aim for are reliability, variability and modeling difficulty. To achieve reliability one major criterion is that the total circuit delay should vary slowly with the elementary component delays. For variability (two chips with different delays have different responses), that same variation should not be too slow. The hardest part, how to create modeling difficulty, remains essentially an open problem.

9. Conclusion

In this paper 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.

Experimental results have shown that there is enough variations between integrated circuits for identification purposes, and that the effect of temperature and power supply voltage variations can be mitigated, allowing robust identification. More experiments are necessary understand the effects of circuit aging on identification ability.

We have argued that delay-based authentication is not susceptible to conventional attacks that attempt to discover a secret, hidden key. The most plausible attack we have found against our devices is the model-building attack. Our experiments suggest that the current implementation of our devices could be vulnerable to this type of attack, but we have suggested a number of mechanisms that might prevent them.

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

REFERENCES

1. R. Anderson and M. Kuhn. Tamper Resistance - a Cautionary Note. In *Proceedings of the Second Usenix Workshop on Electronic Commerce*, pages 1–11, Nov. 1996.
2. R. Anderson and M. Kuhn. Low Cost Attacks on Tamper Resistant Devices. In *IWSP: International Workshop on Security Protocols, LNCS*, 1997.
3. R. J. Anderson. *Security Engineering: A Guide to Building Dependable Distributed Systems.* John Wiley and Sons, 2001.
4. D. S. Boning and S. Nassif. Models of Process Variations in Device and Interconnect. In W. B. A. Chandrakasan and F. Fox, editors, *Design of High Performance Microprocessor Circuits*, chapter 6. IEEE Press, 2000.
5. D. Chinnery and K. Keutzer. *Closing the Gap Between ASIC & Custom.* Kulwer Academic Publishers, 2002.
6. F. Dartu and L. 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.
7. B. Gassend. Physical Random Functions. Master's thesis, Massachusetts Institute of Technology, Jan. 2003.
8. B. Gassend, D. Clarke, M. van Dijk, and S. Devadas. Silicon Physical Random Functions . In *Proceedings of the Computer and Communication Security Conference*, May 2002.
9. B. Gassend, D. Clarke, M. van Dijk, and S. Devadas. Controlled physical random functions. In *Proceedings of* 18^{th} *Annual Computer Security Applications Conference*, December 2002.
10. B. Gassend, D. Clarke, M. van Dijk, and S. Devadas. Delay-based circuit authentication and applications. In *Proceedings of the 2003 ACM Symposium on Applied Computing*, March 2003.
11. P. Kocher, J. Jaffe, and B. Jun. Differential Power Analysis. *Lecture Notes in Computer Science*, 1666:388–397, 1999.
12. Y. Liu, S. R. Nassif, L. T. Pileggi, and A. J. Strojwas. Impact of interconnect variations on clock skew of a gigahertz microprocessor. In *Design Automation Conference*. IEEE/ACM, June 2000.
13. K. Lofstrom, W. R. Daasch, and D. Taylor. IC Identification Circuit Using Device Mismatch. In *Proceedings of ISSCC 2000*, pages 372–373, February 2000.
14. A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography.* CRC Press, 1996.
15. J.-J. Quisquater and D. 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.
16. P. S. Ravikanth. *Physical One-Way Functions.* PhD thesis, Massachusetts Institute of Technology, 2001.
17. 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.
18. S. Theodoridis and K. Koutroumbas. *Pattern Recognition.* Academic Press, 2003.
19. N. Weste and K. Eshraghian. *Principles of CMOS VLSI Design: A Systems Perspective.* Addison Wesley, 1985.