Physical Unclonable Functions for Device Authentication and Secret Key Generation

G. Edward Suh
Cornell University
Ithaca, NY 14853
suh@csl.cornell.edu

Srinivas Devadas
Massachusetts Institute of Technology
Cambridge, MA 02139
devadas@mit.edu

ABSTRACT

Physical Unclonable Functions (PUFs) are innovative circuit primitives that extract secrets from physical characteristics of integrated circuits (ICs). We present PUF designs that exploit inherent delay characteristics of wires and transistors that differ from chip to chip, and describe how PUFs can enable low-cost authentication of individual ICs and generate volatile secret keys for cryptographic operations.

Categories and Subject Descriptors

K.6.5 [**Management of Computing and Information Systems**]: Security and Protection

General Terms

Design, Security

Keywords

IC authentication, Secret keys

1. INTRODUCTION

As electronic devices become ubiquitous and interconnected, people are increasingly relying on integrated circuits (ICs) for performing security sensitive tasks as well as handling sensitive information. For example, an RFID is often used as a key card to control access to buildings, smart cards carry out financial transactions, and mobile phones often contain sensitive data such as confidential documents, personal emails, etc. Therefore, it is critical for ICs to be able to perform operations such as authentication of devices, protection of confidential information, and secure communication in an inexpensive yet highly secure way.

A common ingredient that is required to enable the above security operations is a secret on each IC, which an adversary cannot obtain or duplicate. The current best practice is to place a secret key in non-volatile memory such as fuses and EEPROM, and use cryptographic primitives such as digital signature and encryption to authenticate a device and protect confidential information.

Unfortunately, the conventional approach suffers from a couple of shortcomings. First, safely managing secrets in memory is difficult and expensive. Non-volatile memory technologies are often vulnerable to invasive attack as secrets always exist in a digital form, and even battery-backed RAMs can be read after storing keys for a long time [1, 2, 14]. For a high level of physical security, the IC needs to be protected using expensive tamper-sensing circuitry that needs to be continually battery powered. Second, for extremely resource constrained platforms such as RFIDs, even simple cryptographic operations can be too costly.

Physical Unclonable Functions (PUFs) [5, 6, 7] are innovative primitives to derive secrets from complex physical characteristics of ICs rather than storing the secrets in digital memory. For example, a volatile secret can be generated from the random delay characteristics of wires and transistors. Because the PUF taps into the random variation during an IC fabrication process, the secret is extremely difficult to predict or extract.

PUFs significantly increase physical security by generating *volatile* secrets that only exist in a digital form when a chip is powered on and running. This immediately requires the adversary to mount an attack while the IC is running and using the secret, a significantly harder proposition than discovering non-volatile keys; an invasive attack must accurately measure PUF delays without changing the delays or discover volatile keys in registers without cutting power or tamper-sensing wires that clear out the registers.

This paper discusses how PUFs can enable low-cost authentication of ICs and generate volatile secret keys for cryptographic operations. We also introduce a new PUF circuit design based on ring oscillators, which has advantages in the ease of implementation and reliability over previously proposed designs. The PUF circuits can also be used as hardware random number generators [11]. However, this paper only focuses on device authentication and key generation.

The rest of the paper is organized as follows. Section 2 provides a brief overview of previous work on silicon PUFs, including the concept, a circuit design, and test-chip results. Section 3 describes a different PUF design based on ring oscillators. Section 4 and Section 5 discuss how PUFs can be used for low-cost authentication and cryptographic key generation, respectively. Section 6 demonstrates that the discussed applications are viable by providing experimental results on 90nm FPGAs. Finally, Section 7 compares silicon PUFs to related work and Section 9 concludes the paper.

DAC 2007, June 4-8, 2007, San Diego, California, USA
Copyright 2007 ACM 978-1-59593-627-1/07/0006 ...$5.00.

2. PHYSICAL UNCLONABLE FUNCTIONS

In this section, we introduce the concept of Physical Unclonable Functions (PUFs) and a previously proposed PUF design based on MUXes and an arbiter. We call this an *arbiter PUF*.

2.1 Concept

A Physical Random Function or Physical Unclonable Function (PUF) is a function that maps a set of challenges to a set of responses based on an intractably complex physical system. (Hence, this static mapping is a "random" assignment.) The function can *only* be evaluated with the physical system, and is unique for each physical instance. While PUFs can be implemented with various physical systems (cf. Section 7), this paper focuses on silicon PUFs (SPUFs) that are based on the hidden timing and delay information of integrated circuits [6, 7]. Even with identical layout masks, the variations in the manufacturing process cause significant delay differences among different ICs.

As mentioned in the introduction, PUFs provide significantly higher physical security by extracting secrets from complex physical systems rather than storing them in nonvolatile memory. Another advantage of PUFs is that they do not require any special manufacturing process or programming and testing steps.

2.2 Arbiter PUF

Figure 1 illustrates a silicon PUF delay circuit based on MUXes and an arbiter. The circuit has a multiple-bit input X and computes a 1-bit output Y based on the relative delay difference between two paths with the same layout length. The input bits determine the delay paths by controlling the MUXes. Here, a pair of MUXes controlled by the same input bit $X[i]$ work as a switching box (dotted boxes in the figure). The MUXes pass through the two delay signals from the left side if the input control bit $X[i]$ is zero. Otherwise, the top and bottom signals are switched. In this way, the circuit can create a pair of delay paths for each input X. To evaluate the output for a particular input, a rising signal is given to both paths at the same time, the signals race through the two delay paths, and the arbiter (latch) at the end decides which signal is faster. The output is one if the signal to the latch data input (D) is faster, and zero otherwise.

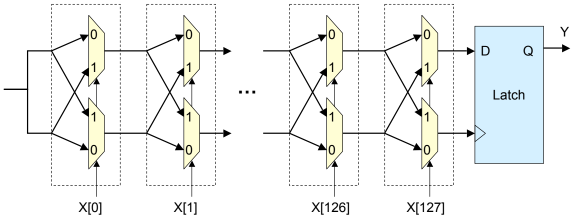


Figure 1: An arbiter PUF delay circuit. The circuit creates two delay paths with the same layout length for each input X, and produces an output Y based on which path is faster.

Because the PUF circuit is rather simple, attackers can try to construct a precise timing model and learn the parameters from many input-output pairs [8]. To prevent these model-building attacks, the PUF circuit output can be obfuscated by XOR'ing multiple outputs or a PUF output can be used as one of the MUX control signals. Note that the model building attack is irrelevant for the cryptographic key generation where the PUF output is never directly exposed.

There are two ways to construct a k-bit response from the 1-bit output of this PUF delay circuit. First, one circuit can be used k times with different inputs. A challenge is used as a seed for a pseudo-random number generator (such as a linear feedback shift register). Then, the PUF delay circuit is evaluated k times, using k different bit vectors from the pseudo-random number generator serving as the input X to configure the delay paths. It is also possible to duplicate the single-output PUF circuit itself multiple times to obtain k bits with a single evaluation.

2.3 Experimental Results

This arbiter-based PUF circuit with 64 stages has been fabricated and tested in TSMC's 0.18μm, single-poly, 6-level metal process [7]. The experimental results show that two identical PUF circuits on two different chips have different outputs for the same input with a probability of 23% (inter-chip variation). On the other hand, multiple measurements on the same chip are different only with 0.7% probability.

Because the circuit measures the relative delay difference, the PUF is robust against environmental variations. For realistic changes in temperature from 20 to 70 Celsius and regulated voltage changes of ±2%, the output noise is 4.8% and 3.7%, respectively. Even when increasing the temperature by 100C and varying the voltage by 33%, the PUF output noise still remains below 9%. This variation is significantly less than the inter-chip variation of 23%, allowing for the identification of individual chips.

An ideally symmetric layout of the circuit in Figure 1 would increase inter-chip variation to 50%. The circuit fabricated on our test chips has systematic skews in the layout because the wires were auto-routed using CAD tools and because of inherent skew in the latch used as an arbiter. Removing the skews with careful layout would increase the inter-chip variation.

3. RING OSCILLATOR PUF

In this section, we introduce a new PUF design based on delay loops (ring oscillators) and counters rather than MUXes and an arbiter. We call this new design an *RO PUF*. Compared to the arbiter PUF described in the previous section, the RO PUF allows an easier implementation for both ASICs and FPGAs, an easier evaluation of the entropy, and higher reliability. On the other hand, the RO PUF is slower, larger and consumes more power to generate bits than the arbiter PUF. Therefore, two designs are complementary; the arbiter PUF is appropriate for resource constrained platforms such as RFIDs and the RO PUF is better for use in FPGAs and in secure processor designs.

3.1 RO PUF Overview

Figure 2 illustrates a PUF delay circuit that is comprised of many *identically* laid-out delay loops (ring oscillators). Each ring oscillator is a simple circuit that oscillates with a particular frequency. Due to manufacturing variation, each ring oscillator oscillates with a slightly different frequency. In order to generate a fixed number of bits, a fixed sequence of oscillator pairs is selected, and their frequencies are compared to generate an output bit. The output bits from the same sequence of oscillator pair comparisons will vary from chip to chip. Given that oscillators are identically laid out, the frequency differences are determined by manufacturing variation and an output bit is equally likely to be one or zero if random variations dominate.

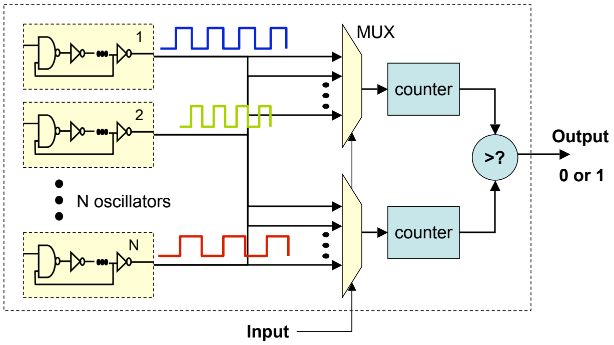


Figure 2: Ring oscillator based PUF circuit.

Note that it is very easy to duplicate a ring oscillator as a hard-macro and ensure that all oscillators are identical. Unlike the arbiter PUF, there is no need for careful layout and routing. For example, the paths from oscillator outputs to counters do not need to be symmetric. By counting many oscillator cycles, the difference in oscillator frequencies can be amplified and will dominate any skews in routing.

Now let us consider how many bits we can generate from this circuit. Each comparison of a pair of oscillators generates a bit. There are $N(N-1)/2$ distinct pairs given N ring oscillators. However, the entropy of this circuit, which corresponds to the number of independent bits that can be generated from the circuit, is clearly less than $N(N-1)/2$ because the bits obtained from pair-wise comparisons are correlated. For example, if oscillator A is faster than oscillator B, the comparison will yield a 1. If B is in turn faster than C, the comparison will yield a 1. It is clear that when A is compared with C that the comparison will yield a 1 - these bits are correlated.

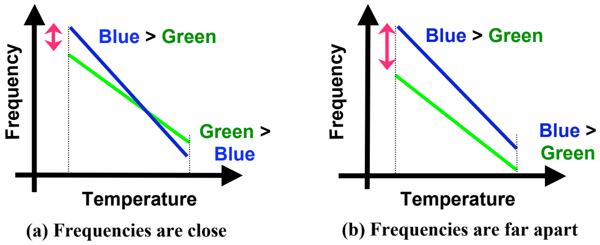
Fortunately, it is possible to derive the maximum entropy of this circuit assuming pair-wise comparisons, i.e., the number of independent bits that can be generated by the circuit as a function of N, the number of oscillators. There are $N!$ different orderings of ring oscillators based on their frequencies. If the orderings are equally likely, the entropy will be $log_2(N!)$ bits. For example, 35 oscillators can produce 133 bits, 128 oscillators can produce 716 bits, and 1024 oscillators can produce 8769 bits.

For simplicity, it is also possible to use each oscillator only once to generate a single bit and avoid any correlation. For example, 128 pairs of oscillators (256 oscillators total) can be used to generate 128 independent bits.

3.2 Reliability Enhancement

Ring oscillator frequencies change significantly as environmental conditions such as temperature and voltage change. Of course, we are not using absolute frequencies but rather doing relative comparisons. The PUF output changes only if the ordering of the two oscillators being compared changes.

Figure 3 shows how errors ("bit-flips") could occur due to environmental changes. Say that ring oscillator Blue is faster than ring oscillator Green at room temperature. However, when the temperature increases, both oscillators slow down, with Blue slowing down faster than Green, due to different device or physical parameters. These ring oscillators "flip" when the temperature changes substantially. This flip causes an error in the generated bit.



(a) Frequencies are close

(b) Frequencies are far apart

Figure 3: The relationship between the ring oscillator frequency distance and the probability of a PUF output flip.

The insight from Figure 3 is that ring oscillators whose base frequencies are far apart are much less likely to flip than ring oscillators whose frequencies are close together. This insight can be used to dramatically reduce the error rate of generated key bits by judiciously selecting ring oscillator pairs that will be compared. Specifically, we can remove a significant portion of errors if we only compare ring oscillator pairs, whose frequencies are far apart, to generate key bits.

As an example, we will show results from a very simple 1-out-of-k masking scheme with $k = 8$ in Section 6. As mentioned earlier, a fixed sequence of ring oscillator pairs is generated, this sequence now needs to be k times longer than the desired number of bits to be generated. Then, for each k ring oscillator pairs, we choose the pair that has maximum distance. The bit vector indicating these selections is saved so that the same pairs can be used to re-generate the output. Other masking schemes such as picking n out of m, or using a distance threshold are also possible.

There is a very small probability that key bits will have errors once this masking is performed. Depending on the application, the remaining errors can be corrected using an error correcting code or merely forgiven (e.g., the chip is authenticated if the number of errors is very small).

3.3 Creating Challenge-Response Pairs

For low-cost authentication shown in the next section, the PUF circuit must be able to produce exponentially-many challenge-response pairs. Unfortunately, the RO PUF that was discussed can only generate a relatively small number of bits. There are a few ways to create many challenge-response pairs. First, we can extend the oscillators to have configurable delay paths similar to the one in the arbiter PUF shown in Figure 1. A challenge selects how to configure the path within a delay loop so that different challenges result in a different oscillation frequency. Second, in programmable logic such as FPGAs, a challenge can determine the oscillator configuration such as the number of inverters and which look-up tables and wires to be used. Each challenge will create a PUF circuit using different parts of the logic, resulting in a unique response.

4. LOW-COST AUTHENTICATION

This section discusses how PUFs can be used to authenticate individual ICs without costly cryptographic primitives. The PUF-based authentication described here can be applied even to extremely resource constrained platforms such as RFIDs where cryptographic operations may be too expensive in terms of silicon area or power consumption, or off-the-shelf programmable ICs such as FPGAs where cryptographic operations are not implemented. In the next section, we also discuss how PUFs can generate cryptographic keys that can be used for any cryptographic operations including data encryption and authentication.

As the PUF output is unique and unpredictable for each IC, provided it is long enough, it is straightforward to identify an IC with the PUF. (See Section 7.) One can simply record a PUF output and compare that with a re-generated one later. However, a single PUF output per IC is not enough for authentication; anyone who has access to an IC can obtain the PUF output and create another IC that contains the PUF output in memory. Therefore, the authentication mechanism should ensure that an adversary cannot obtain the PUF output that is used for authentication.

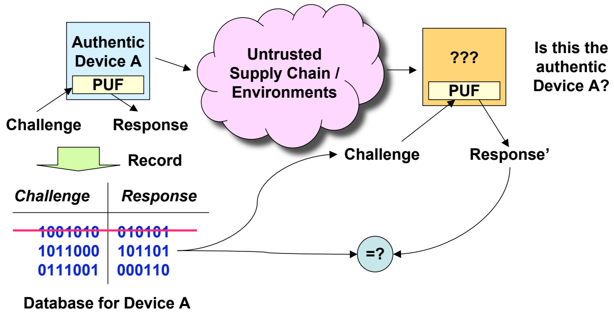


Figure 4: Overview of PUF-based authentication.

Figure 4 illustrates the PUF-based authentication process. Here, we exploit that the PUF can have exponential number of challenge response pairs where the response is unique for each IC and each challenge. We also assume that model-building is hard to do for a given PUF, because of non-linearities in the PUF. (For an analysis of model-building on arbiter PUFs see [8].) A trusted party, when in possession of an authentic IC, applies randomly chosen challenges to obtain unpredictable responses. The trusted party stores these challenge-response pairs in a database for future authentication operations. To check the authenticity of an IC later, the trusted party selects a challenge that has been previously recorded but has never been used for an authentication check operation, and obtains the PUF response from the IC. If the response matches (i.e., is close enough to) the previously recorded one, the IC is authentic because only the authentic IC and the trusted party should know that challenge-response-pair. To protect against man-in-the-middle attacks, challenges are never reused. Therefore, the challenges and responses can be sent in the clear during authentication operations.

5. CRYPTOGRAPHIC KEY GENERATION

In many security applications, cryptographic primitives such as encryption, digital signatures, and message authentication code play central roles. Unfortunately, outputs from the PUF circuits as described are inappropriate as cryptographic keys. Because of noise, the outputs are likely to be slightly different on each evaluation, even on the same IC for the same challenge. On the other hand, cryptographic primitives require that every bit of a key stays constant. Moreover, some primitives such as RSA require keys to satisfy specific mathematical properties whereas the PUF outputs are randomly determined by manufacturing variations.

In this section, we discuss how PUFs can generate volatile secret keys that can be used for cryptographic operations. There are two components. First, the error correction process, which consists of initialization and re-generation, ensures that the PUF can consistently produce the same output even if there are significant environmental changes such as voltage and temperature fluctuations. Second, the key generation process converts the PUF output into cryptographic keys. The overall process is shown in Figure 5.

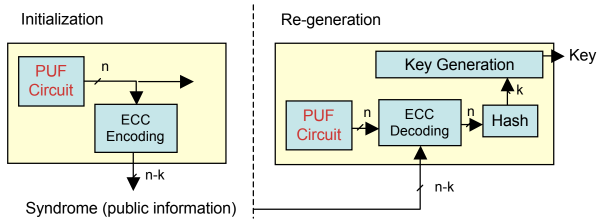


Figure 5: Cryptographic key generation with PUFs.

In the initialization step, an output is generated from the PUF circuit and the error correcting syndrome for that output is computed and saved for later. For example, the BCH code can be used to compute the syndrome. The syndrome is information that allows for correcting bit-flips in re-generated PUF outputs. If a masking scheme is used with an RO PUF as described in Section 3.2, the bit vector that selects oscillator pairs will also be stored along with the syndrome. Note that the syndrome and this bit vector are public information and can be stored anywhere (on-chip, off-chip, or remotely on a server).

To re-generate the same PUF output, the PUF first produces an output from the circuit. If there is a saved bit vector, then that is used to select pairs. Then, the PUF uses the syndrome from the initialization step to correct any changes in the circuit output. In this way, the PUF can consistently reproduce the output from the initialization step.

Clearly, the syndrome reveals information about the PUF delay circuit output. In general, however, given the b-bit syndrome, attackers can learn at most b bits about the PUF delay circuit output. Therefore, to obtain k secret bits after the error correction, we generate $n = k + b$ bits from the PUF delay circuit. Even with the syndrome, an adversary still needs to guess at least k bits to find the correct PUF response. For example, we can use the BCH (127,64,21) code to reliably generate 64-bit secrets. The BCH (n, k, d) code can correct up to $(d-1)/2$ errors out of n bits with an $(n-k)$-bit syndrome ($b = n - k$).

While the mask may reveal information about what ring oscillator frequencies are far apart, it does not reveal information about the sign of comparisons, i.e., the bits generated. If a ring oscillator is used many times to generate bits, then it is conceivable that information about ordering of ring oscillators can be extracted from the mask. This can be easily precluded by using each oscillator only once to generate a single bit.

For cryptographic operations that use a randomly selected number as a key, the output of the error correcting code (ECC) can be simply hashed down to a desired length and used as a cryptographic key. For example, symmetric key primitives such as AES can use the hashed PUF output.

For cryptographic operations whose keys need to satisfy special properties (for example, an RSA key pair), the hashed PUF output is used as a seed for a key generation algorithm. In this way, the PUF can generate keys for any cryptographic operation. We note that PUFs simply generate keys that can be used with standard algorithm. There is no change required in cryptographic algorithms.

The PUF with its key generation capability can be tightly integrated with a processor to enable a physically secure processor [16].

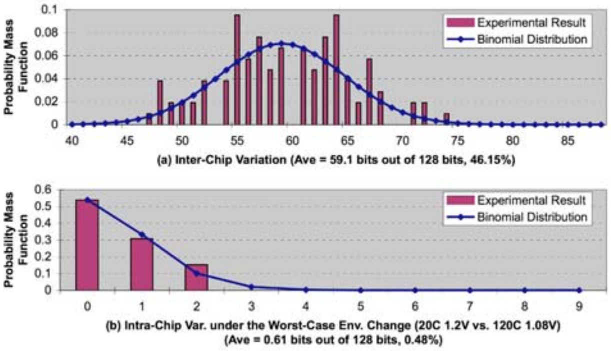
6. EXPERIMENTAL VALIDATION

Section 2.3 briefly discussed the experimental results for an ASIC implementation of the 64-stage arbiter PUF. This section presents recent experimental results for the RO PUF on FPGAs and shows that the PUF can be used for authentication as well as secret key generation.

The experiments in this section are performed on 15 Xilinx Virtex4 LX25 FPGAs (90nm). All 15 FPGAs are exactly the same model, and therefore identical designs. We placed 1024 ring oscillators in each FPGA as a 16-by-64 array. Each ring oscillator consists of 5 inverters and 1 AND gate, which are implemented using look-up tables (LUTs). To generate reliable outputs, the 1-out-of-8 mask scheme described in Section 3.2 is used.

In order to evaluate the basic PUF functions, we need to know whether the PUF outputs are *unique* (for security) and *reproducible* (for reliability). We define the following two metrics for this purpose.

- *Inter-chip variation*: How many PUF output bits are different between PUF A and PUF B? This is a measure of uniqueness. If the PUF produces uniformly distributed independent random bits, the inter-chip variation should be 50% on average.
- *Intra-chip (environmental) variation*: How many PUF output bits change when re-generated again from a single PUF with or without environmental changes? This indicates the *reproducibility* of the PUF outputs. Ideally, the intra-chip variation should be 0%.



(a) Inter-Chip Variation (Ave = 59.1 bits out of 128 bits, 46.15%)

(b) Intra-Chip Var. under the Worst-Case Env. Change (20C 1.2V vs. 120C 1.08V) (Ave = 0.61 bits out of 128 bits, 0.48%)

Figure 6: The probability distributions for the inter-chip variation and the intra-chip variation with the worst case environmental change.

Figure 6 (a) illustrates the probability distribution of the inter-chip variation when 128 bits are produced from each PUF (FPGA). The x-axis represents the number of PUF output bits that are different between two FPGAs, and the y-axis represents the probability. Here, the bars (purple) show the experimental results from 105 pair-wise comparisons, and the line (blue) shows a binomial distribution with parameters fitted to the experimental results ($n = 128$ and $p = 0.4615$). As shown in the figure, we obtain the average inter-chip variation of 46.15%, which is pretty close to the ideal average of 50%.

Figure 6 (b) illustrates the reproducibility of PUF outputs by showing the intra-chip variation with the worst-case environmental change. For each FPGA, PUF outputs are generated at two different environmental conditions and compared. Here, changing the temperature from 20C to 120C and the core voltage from 1.2V to 1.08V changed the PUF output by 0.6 bits on average (0.48%). This results show that the intra-chip variation is much lower than the inter-chip variation even in the worst-case environmental change. While we studied the intra-chip variation under various temperature (-20C to 120C) and voltage (+/-10%) changes, the figure only shows the case that resulted in the highest intra-chip variation. In fact, the PUF outputs did not change at all for small or moderate environmental changes.

In summary, the experiments show that two identical PUF circuits on two different FPGAs produce a different output bit with a probability of 46.15% on average. On the other hand, multiple measurements on the same chip are different only with 0.48% probability (i.e., 0.6 bits out of 128) even in the worst-case environmental change. Also, while the number of samples are small, the figure demonstrates that the binomial distributions match the experimental results fairly well.

From the inter-chip and intra-chip variations, we can compute false positive and false negative rates when the PUF outputs are used in the simple authentication scheme described in Section 4.

- *False positive rate*: The probability that PUF A will be authenticated as PUF B; PUF A produces the same output with PUF B.
- *False negative rate*: The probability that a correct PUF will fail to be authenticated; the PUF fails to re-generate a consistent output.

If we allow up to 10 bits out of 128 bits to be different in order for an IC to be authenticated, the above results indicate that the false positive rate is about 2.1×10^{-21} and the false negative rate is less than 5×10^{-11}. In this analysis, we assume that the inter-chip and intra-chip variations follow binomial distributions.

Similarly, the same analysis can be applied to estimate how reliable the PUF-generated cryptographic keys will be. For example, if the BCH (127,64,21) code is used, 10 errors in a 127-bit PUF output can be corrected and the probability of failing to re-generate the same key is less than 5×10^{-11}.

7. RELATED WORK

7.1 IC Identification

It is widely known that significant process variations exist in IC fabrication making each IC unique [3, 4, 10]. These process variations have previously been used to identify ICs [9] by exploiting fluctuations in drain currents. However, this identification method is not secure as an adversary can easily read out the ID and duplicate an IC. Moreover, these circuits cannot produce reliable bits that can be used with cryptographic operations.

7.2 Other Types of PUFs

Researchers have studied the implementation of PUFs exploiting physical characteristics other than timing and delay information of silicon circuits. For example, Pappu proposed an optical PUF, which uses the speckle patterns of optical medium for laser light [12]. Coating PUFs and acoustic PUFs [13, 17] measure the capacitance of a coating layer covering an IC and the acoustic reflections of a token, respectively. This paper focuses on silicon PUFs which are very easy to integrate into ICs and processors unlike other types of PUFs.

7.3 Memory Technologies to Store Secrets

Conventional solutions to embed secrets in ICs and computing devices rely on battery-backed RAM or non-volatile memories such as ROMs, fuses, or flash / EEPROM. When using these technologies, on-chip memory is programmed with a key that needs to stay secret.

PUFs provide unique secrets for each IC with stronger physical security. Also, unlike keys in memory, PUFs do not require programming of secrets and are difficult to duplicate even by an IC manufacturer. Finally, because the PUF circuit only uses standard digital logic, there is no need for additional mask or special fabrication steps and PUFs can be easily scaled to advanced technologies.

7.4 Tamper Sensing Packages

The state of the art method for protecting against key extraction through invasive physical attack is to enclose the secret key in a tamper sensing package as in the IBM 4758 processor [15, 18]. This type of protection provides a high level of security, but is expensive and must be continuously powered.

8. ACKNOWLEDGEMENT

We thank Tom Ziola of PUFCO, Inc. for his support of this project. We thank Daihyun Lim, Jaewook Lee, Blaise Gassend, Dwaine Clarke and Marten van Dijk for their contributions to the PUF project at MIT.

9. CONCLUSION

We have described Physical Unclonable Functions (PUFs) and showed PUFs can provide low-cost authentication of ICs and generate volatile keys for both symmetric and asymmetric cryptographic operations. Ongoing work includes the implementation of PUF-enabled RFIDs and the development of a secure processor that uses a PUF to generate cryptographic keys that are only known to the processor.

10. REFERENCES

[1] R. Anderson and M. Kuhn. Tamper resistance - a cautionary note. In *Proceedings of the 2^{nd} USENIX Workshop on Electronic Commerce*, November 1996.

[2] R. Anderson and M. Kuhn. Low cost attacks on tamper resistant devices. In *IWSP: International Workshop on Security Protocols*, 1997.

[3] D. S. Boning and J. E. Chung. Statistical metrology: Understanding spatial variation in semiconductor manufacturing. In *Proceedings of SPIE 1996 Symposium on Microelectronic Manufacturing*, 1996.

[4] K. A. Bowman, S. G. Duvall, and J. D. Meindl. Impact of die-to-die and within die parameter fluctuations on the maximum clock frequency distribution for gigascale integration. *Journal of Solid-State Circuits*, 37(2):183–190, February 2002.

[5] 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.

[6] B. Gassend, D. Clarke, M. van Dijk, and S. Devadas. Silicon physical random functions. In *Proceedings of the Computer and Communication Security Conference*, November 2002.

[7] J.-W. Lee, D. Lim, B. Gassend, G. E. Suh, M. van Dijk, and S. Devadas. A technique to build a secret key in integrated circuits with identification and authentication applications. In *Proceedings of the IEEE VLSI Circuits Symposium*, June 2004.

[8] D. Lim. Extracting secret keys from integrated circuits. Master's thesis, Massachusetts Institute of Technology, May 2004.

[9] K. Lofstrom, W. R. Daasch, and D. Taylor. Ic identification circuit using device mismatch. In *Proceedings of ISSCC 2000*, February 2000.

[10] S. R. Nassif. Modeling and forecasting of manufacturing variations. In *Proceedings of ASP-DAC 2001*, 2001.

[11] C. W. O'Donnell, G. E. Suh, and S. Devadas. PUF-based random number generation. In *MIT CSAIL CSG Technical Memo 481*, November 2004.

[12] R. Pappu. *Physical One-Way Functions*. PhD thesis, Massachusetts Institute of Technology, 2001.

[13] B. Skoric, P. Tuyls, and W. Ophey. Robust key extraction from physical unclonable functions. In *Proceedings of the Applied Cryptography and Network Security Conference 2005*, volume 3531 of *Lecture Notes in Computer Science*, 2005.

[14] S. P. Skorobogatov. Semi-invasive attacks - a new approach to hardware security analysis. In *Technical Report UCAM-CL-TR-630*. University of Cambridge Computer Laboratory, April 2005.

[15] S. W. Smith and S. H. Weingart. Building a high-performance, programmable secure coprocessor. *Computer Networks (Special Issue on Computer Network Security)*, 31(8):831–860, April 1999.

[16] G. E. Suh, C. W. O'Donnell, I. Sachdev, and S. Devadas. Design and implementation of the aegis single-chip secure processor using physical random functions. In *Proceedings of the 32^{nd} Annual International Symposium on Computer Architecture*, June 2005.

[17] P. Tuyls, B. Skoric, S. Stallinga, A. Akkermans, and W. Ophey. Information theoretical security analysis of physical unclonable functions. In *Proceedings of Conference on Financial Cryptography and Data Security 2005*, volume 3570 of *Lecture Notes in Computer Science*, 2005.

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