

Article **Key Generation for Hardware Obfuscation Using Strong PUFs**

Md Shahed Enamul Quadir [*](https://orcid.org/0000-0003-3807-5465) and John A. Chandy

Department of Electrical and Computer Engineering, University of Connecticut, Storrs, CT 06269-4157, USA ***** Correspondence: md.enamul_quadir@uconn.edu

Received: 8 May 2019; Accepted: 13 July 2019; Published: 17 July 2019

Abstract: As a result of the increased use of contract foundries, intellectual property (IP) theft, excess production and reverse engineering are major concerns for the electronics and defense industries. Hardware obfuscation and IP locking can be used to make a design secure by replacing a part of the circuit with a key-locked module. In order to ensure each chip has unique keys, previous work has proposed using physical unclonable functions (PUF) to lock the circuit. However, these designs are area intensive. In this work, we propose a strong PUF-based hardware obfuscation scheme to uniquely lock each chip.

Keywords: obfuscation; key generation; authentication; physically unclonable function; counterfeiting

1. Introduction

Security has become a major concern for integrated circuits (ICs) because of globalization and outsourced offshore production. Throughout the lifecycle of the IC, the chip could be under attack from overproduction during the manufacturing stage to unauthorized recycling after customer disposition [\[1\]](#page-8-0). Furthermore, a manufacturer could make extra chips by cloning during the manufacturing stage after obtaining the design by reverse engineering [\[2](#page-8-1)[,3\]](#page-9-0). State of the art IC reverse engineering is so advanced that the chips could be reverse engineered within a few weeks. There are dedicated companies that do reverse engineering of new industrial chips. Therefore, a scheme is required to prevent the production of illegal cloned chips by reverse engineering.

Hardware obfuscation is an approach to prevent IC piracy and reverse engineering. Hardware obfuscation could be categorized into two types: Logic or functional locking and camouflage. The main idea behind logical locking obfuscation is that part of the design is replaced with a configurable module at the design stage. If the module is not activated by the designer, the chip will not function properly [\[4](#page-9-1)[–7\]](#page-9-2). During the post-fabrication activation process in a trusted design house, the chips can be activated by unlocking the obfuscated function with a secret key that may be burned into on-chip fuses. Those unlocked chips can then be sold to the open market. The stored key can not be recovered without direct access to the on-chip fuses such as with probing attacks. Therefore, an attacker can not reverse engineer the design because of the obfuscation, and the chip can not be overproduced without knowledge of the key. Furthermore, layout level techniques such as cell camouflage [\[8\]](#page-9-3) could be used as hardware obfuscation and dummy contacts are used to protect against attackers. The layout of standard cells with different functionalities is made to appear identical in the camouflage technique. Camouflaging can make it more difficult to identify camouflaged gates with automated image tools. In this paper, we have focused on logic locking-based hardware obfuscation.

Several conceptually new and interesting approaches have been proposed to introduce forms of obfuscation or logic locking to prevent reverse engineering and piracy. For example, additional exclusive OR (XOR) gates can be inserted in combinational designs with configurable bits that need to set correctly to activate the chips [\[6\]](#page-9-4). Dummy states can be inserted into finite state machines in sequential designs, requiring certain sequences of inputs to be applied for the circuits to function correctly [\[9\]](#page-9-5). Scrambling the interconnection network in a permutation-based technique will cause the circuit to only work with a correct key which will configure the interconnection network [\[10\]](#page-9-6). Other work has hidden parts of the circuit in a configurable lookup table (LUT) [\[11\]](#page-9-7). With those obfuscation methods, the effort for an attacker to find the correct key will become computationally infeasible [\[12\]](#page-9-8). The assumption made with these methods is that there is no direct access to the key content [\[6,](#page-9-4)[13\]](#page-9-9).

A problem with many of these approaches is that the key is not unique across all instances of the chip. Thus, if an attacker is able to retrieve the key by some method, it can unlock all chips and effectively overproduce the chips. Recently, physically unclonable functions (PUFs) have been used as a way to provide unique keys for obfuscation [\[14](#page-9-10)[–16\]](#page-9-11). PUFs are inherent circuit primitives that extract randomness from the physical characteristics of a system at the manufacturing stage [\[17](#page-9-12)[,18\]](#page-9-13) by applying input challenges and observing random output responses. PUFs are easy and simple to implement but their random nature ideally makes its behavior hard to predict and model for an attacker.

Some of these PUF-based obfuscation approaches require a full characterization of the PUF input-output response, thereby necessitating a relatively small output space for the PUF. Such PUFs are called weak PUFs. An untrusted foundry could do the same characterization and could store the challenge-response pairs (CRP) for all chips before sending them to the design house. If a key leaked from an activated chip, the untrusted foundry could recover the entire design with the leaked key and, using its PUF characterizations, be able to unlock all chips whether authorized or not.

Strong PUFs, on the other hand, have a very large input/output space, making characterization impractical and thus much more secure. However, at the same time, it makes some PUF-based obfuscations approaches infeasible. It would be ideal to have an obfuscation scheme that can take advantage of these strong PUFs to improve the security of the approach. In this paper, we have proposed using a strong PUF with a subset of bits used to generate a key that can unlock a locked circuit. A mathematical probability model is developed which determines the feasibility of the PUF key.

In Figure [1,](#page-1-0) the design and fabrication flow is shown for a prevention method of piracy and overproduction of chips using a PUF-based method. The designer uses logic synthesis, technology mapping, and place and route tools to produce a graphic database system II (GDSII) file that can be sent to the fabrication facility. Fabricated chips will be manufactured and tested at the foundry and will be sent back to the design house for activation. The design house will activate the chip using the PUF-based key and will send to the market for sale.

The rest of this paper is organized as follows. First, we describe prior related work. Second, we provide in detail a description of the proposed PUF-based hardware obfuscation method. Next, we describe attack analysis for our proposed method. After that we analyze our technique based on performance analysis. And finally, we evaluate our technique on different ISCAS 85 benchmarks.

Figure 1. Design and fabrication flow of the physical unclonable functions (PUF)-based obfuscation.

2. Prior PUF-Based Obfuscation Approaches

A PUF is an unclonable hardware function based on process variation during fabrication. A PUF has a set of challenge response pairs (CRP) which are the input and output respectively. The CRPs represent a random function which is unique to each chip. It is impossible to make identical PUFs with the same CRPs [\[18\]](#page-9-13). PUFs have been used in different applications including secret key generation, device authentication, and anti-counterfeiting [\[19\]](#page-9-14). Below, we describe prior weak PUF and strong PUF-based approaches to prevent reverse engineering and piracy.

Wendt et al. [\[15\]](#page-9-15) created a unique key per chip using a weak PUF. A part of the circuit's functionality was replaced with a PUF and a lookup table (LUT). The functionality of the PUF and the LUT together was equivalent to the original circuit functionality. The content of the LUT for each chip was determined by the designer after obtaining the PUF's CRPs. Thus, the LUT behaved as a key that would be uniquely configured by the designer in the post-fabrication activation process. A problem with this method was that an untrusted manufacturer could have access to the PUF during the fabrication process. The manufacturer could obtain the CRPs of the weak PUF easily and store those results. If the attacker could find a leaked LUT key of an activated chip in the aftermarket, in combination with the previously stored PUF functionality of the leaked chip, it was trivial to then determine LUT keys for any device.

Khalegi et al. [\[14\]](#page-9-10) used a similar approach but with a strong PUF with a large CRP space. In order to make the use of a strong PUF feasible, the designer's characterization of the PUF was limited to just a subset of the input set. For example, if the input of the PUF is *p* bits long, then only *n* bits were used for characterization by the designer, where *n* << *p*. However, an attacker would have to still characterize the entire *p*-bit input PUF. An obfuscator circuit was used to select the *n* input bits. The added complexity of an obfuscator circuit and selection bits added to the overhead. Moreover, the approach does not scale well because as *n* grows, the complexity of the LUT grows exponentially. Thus, one can not obfuscate large circuits, but only small subsections of the circuit. Furthermore, another limitation of this approach is that it does not account for collision of PUF outputs—i.e., with small *n* it is likely that different inputs could produce colliding PUF outputs, which means it would be impossible to realize a LUT that will satisfy the design functionality. To overcome those limitations, we similarly propose a hardware obfuscation method that does not require the full characterization of a strong PUF and has significantly less overhead.

Alkabani et al. [\[20\]](#page-9-16) used a strong PUF (or RUB as it is called in the paper) to generate bits that could then be transformed into a code to unlock the FSM-based locking scheme. Specifically, a specific input external key was applied as a challenge to the PUF response and XOR'd with a separate external key to generate the appropriate internal key that unlocks the FSM. Note that we refer to "internal key" as the internal bits that unlock the locked circuit, while "external key" is the set of bits that are applied externally by a user or from stored memory. Our approach also uses a strong PUF to generate responses based on an input challenge. However, the distinction is that we allow the PUF response to directly create the internal key (or sub-key). In other words, we apply multiple challenges where each response forms part of the internal key. The external key is the set of challenges that are used to create the internal key. This simplifies the circuitry since the PUF output does not need to be as large as the internal key and moreover removes the need for any XOR circuitry.

3. Strong PUF-Based Key Generation

The main idea of the proposed methodology is to eliminate the use of a LUT for obfuscation and use one of the many existing key-based hardware obfuscation methods [\[21\]](#page-9-17). These methods have much lower overheads than LUTs for large *n*. The PUF-based obfuscation architecture is shown in Figure [2.](#page-3-0) The key will be directly generated from a strong PUF given the appropriate challenge. The key can then be provided to any key-based obfuscation method. We assume a PUF with *p* inputs and *n* outputs where *p* >> *n*. One can use an obfuscator circuit to limit the input set, but as we will show, that is not necessary for large *n*. Large *n* can be problematic in terms of characterization, but for our approach a full characterization is not necessary.

Assume that the PUF generates an *n*-bit response given a *p*-bit challenge. As the designer, instead of characterizing the entire *p*-bit input space of the PUF, we only need to find one PUF input that matches the desired key of the locked circuit. Assuming a truly random PUF, on average, that will take 2^{*n*} brute force tries, rather than the 2^{*p*} challenges needed to characterize the entire PUF. As *n* gets

large, however, even that can become difficult. Therefore, instead of using a *n*-bit PUF, we use only *n* 0 bits of the PUF output, where $n' < n$. In order to generate the key, we issue $S = \frac{n}{n'}$ challenges, where each response is stored in a *n'*-bit sub-key register. The *S* sub-keys together form the *n*-bit key *K*:

$$
K = PUF(C_1)|PUF(C_2)|...|PUF(C_S)
$$
\n⁽¹⁾

An attacker would need to apply *S p*-bit challenges in the correct order to find the right key to unlock the device—in other words they would need to guess 2^{Sp} challenges.

The flow diagram for PUF-based obfuscation is shown in Figure [3.](#page-3-1) Our goal is to generate a specific obfuscation key (*K*) from a set of challenge bits (*C*) that is unique to a specific chip as shown in Figure [3a](#page-3-1). Internally, the key generation block consists of a PUF and sub-key registers as described earlier (Figure [3b](#page-3-1)). The sub-keys which will be used for the final key which will be configured by the designer during the post-fabrication activation shown in Figure [3c](#page-3-1). To configure the sub-keys, the designer obtains the CRPs from the PUF during the activation process using the characterization channels Figure [3c](#page-3-1). After characterization, the designer should remove the channels (Figure [3d](#page-3-1)) so that the PUF cannot be probed after market. Techniques to remove the characterization channels include laser burning the accessing wires and burning the supporting fuses, amongst other approaches.

Figure 2. Overview of PUF-based logic encryption.

Figure 3. Design flow for PUF-based obfuscation. (**a**) The key is generated for a part of a design using *p* inputs and *n* outputs. (**b**) The key generation process uses a strong PUF to create sub-keys which comprise the master key. This chip will be only functional when the designer activates it. (**c**) The designer configures the sub-keys based on challenge-response pairs (CRPs) from the PUF and the final key from the sub-keys. (**d**) The activated chip is in the open market.

3.1. Challenge Selection Process

Before a chip can be activated, the designer must select the challenges required to generate the PUF responses/sub-keys that can be combined to create the obfuscation key. As mentioned above, we have divided the *n*-bit key into *S n'*-bit sub-keys as shown in more detail in Figure [4.](#page-4-0) The sub-keys will be visible to the designer and manufacturer but not to the end user. In order to find a challenge that generates a particular sub-key, we need to try, on average, 2^{n'} challenges. Since we have to find S sub-keys, we need to find *S* challenges (assuming no repetitions in the sub-keys).

Figure 4. Sub-key generation for the proposed method.

How many challenges do we need to try to find these *S* sub-keys? Or, put another way, given that we will attempt *N* challenges, what is the probability we find the *S* sub-keys? The *N p*-bit challenges generate *N n'*-bit responses. Assume that *N* − *i* of the responses do not contain any of the required *S* sub-keys, or in other words, *i* responses contain all of the *S* sub-keys and only the *S* sub-keys (with repetition). There are $\binom{N}{i}$ possible positions for those *i* responses. Those *i* responses can be partitioned into *S* buckets in $\{\frac{i}{S}\}$ *S*! ways, where $\{\frac{i}{S}\}$ is the Stirling number of the second kind:

$$
\begin{Bmatrix} i \\ S \end{Bmatrix} = \frac{1}{S!} \sum_{j=0}^{S} (-1)^{S-j} \binom{S}{j} j^{i}.
$$
 (2)

The remaining $N - i$ responses have $(2^{n'} - S)^{N-i}$ possibilities. Given that there are in total $(2^{n'})^N$ possible responses to the *N* challenges, we can put it all together to arrive at the probability that we find the *S* sub-keys:

$$
\sum_{i=S}^{N} \frac{\binom{N}{i} \left\{ \frac{i}{S} \right\} S! (2^{n'} - S)^{N-i}}{(2^{n'})^N}.
$$
\n(3)

Using a similar analysis to the coupon collector problem, the expected value for *N* can be calculated as:

$$
E[N] = \sum_{j=1}^{S} \frac{2^{n'}}{j}.
$$
\n(4)

For example, when $n = 128$, $n' = 8$, and $N = 4000$, the probability that you will get all 16 sub-keys is 0.999997. The expected number of challenges we need to get the 16 sub-keys is 865. When $n = 256$ and $n' = 16$, the expected number of challenges to find the 16 sub-keys we need is 221,559—not an unreasonable number. As n' grows, the expected value grows exponentially. For $n = 256$ and $n' = 32$, the expected value is 11.6×10^9 . At 10 ns per challenge, it would take roughly two minutes to find the required sub-keys. A larger *n'* provides security against attackers who have advanced probing capabilities and have visibility to the PUF responses and can thus limit their challenge tries to *S*2 *n* 0 rather than 2^{Sp} for those that don't. However, if that is not a concern, a smaller *n'* is more practical.

3.2. Chip Activation

Chip activation could be accomplished in a number of ways. In one approach, once the challenges have been selected by the designer, they can be programmed into one-time-programmable ROMs and fused in permanently. This allows the designer to control the activation process. Fuses, however, can be reverse engineered, so care must be taken to have these sub-keys programmed in such a way that cannot be recovered. Alternatively, end-user customers can activate the chip directly with activation challenges provided by the designer. These challenges would be unique to each chip, but since the challenges are not permanently programmed into the chip, they would need to be reapplied on every powerup. This approach allows users to effectively disable a chip until some trusted hardware or software has enabled the chip with the appropriate challenges. There may be scenarios where chips are shipped directly from manufacturer to customer. In this case, the PUF characterization channel will need to be left open. The customer could then apply several challenges and get the associated responses for the key. These could be sent to the designer, who can select the specific challenges that will generate the activation key. After activation, an automated mechanism can disable the characterization channel so that those channels can not be used in future. This approach opens up more attack possibilities—for example, end-users can now use machine learning attacks to characterize the PUF, and with known "good" responses, they can possibly recover the key. Therefore, this last activation method should be used only with trusted customers.

4. Security Analysis

4.1. Manufacturing Attacks

An untrusted manufacturer has access to the PUF input/output (I/O) channels, and could find the challenges required to generate the sub-keys in the same way as the designer and not require all 2^{Sp} challenges. The manufacturer can then unlock overproduced chips assuming that it can get access to a leaked obfuscation key. If the manufacturer has access to the locked netlist, Boolean satisfiability problem (SAT)-based attacks could be used to recover the sub-keys. Once the key has been retrieved, the manufacturer could then unlock chips, or even modify the design to remove the PUF completely and apply the retrieved key directly. Therefore, the underlying obfuscation method must be SAT-resistant. Since our proposed technique will work with any key-based obfuscation technique, as more promising anti-SAT obfuscation technologies are developed [\[21\]](#page-9-17), our technique will work as well.

4.2. In the Field Attacks

We assume, for the purposes of this discussion, that the in-the-field chips no longer have the PUF characterization channels open. As mentioned before, the activation method that leaves PUF characterization channels open to the end-user is only appropriate with fully trusted customers. Unlike other key-based hardware obfuscation approaches, in our mechanism, the raw key is never applied directly to the circuit. A set of challenges will only unlock a specific chip and will not reveal any information about the underlying obfuscation key. Thus, the chances of a leaked obfuscation key are greatly reduced. An attacker must try 2^{Sp} challenges to guess the key and those set of challenges, if guessed correctly, would only work on that particular chip, so brute force attacks are not practical.

It is possible that advanced probing attacks could reopen the PUF characterization channels and allow an attacker to generate sub-keys easily. As mentioned above, this only works if the underlying obfuscation key is known. Even if the key is known, the set of challenges works only on one chip and it is not practical to reopen the PUF channels on all chips and perform the attack at scale. One could potentially use machine-learning attacks that can fully characterize the PUF with a mathematical model [\[22](#page-9-18)[,23\]](#page-9-19). Although these attacks can be used to model a large variety of strong PUFs with high accuracy and precision in a reasonable time, there are a number of optimizations that have been proposed that increase the modeling time significantly such that it is impractical for an attacker to perform these attacks at scale. It should be noted that the same probing techniques could retrieve the key from LUT-based schemes as well.

5. Performance Analysis

5.1. Hardware Complexity

For our obfuscation key generation, we will need a strong PUF with *p* inputs and *n'* outputs. Most strong PUF implementations have $O(p + n')$ complexity. Other than the PUF, the primary costs to our obfuscation key generation are the sub-key registers and some simple control logic to load the sub-key registers appropriately. We need *n* flip-flops for the sub-keys and control logic is roughly *O*(log *S*). Most key-based obfuscation schemes are *O*(*n*) with respect to overhead. Compared to a LUT-based scheme, which is $O(2^n)$ in size complexity, our key generation plus obfuscation has significantly less overhead.

5.2. Uniqueness and Reliability of the PUF

It is possible that some input challenges could produce unstable output for the PUF at different operating conditions such as temperature variations, and voltage fluctuations. Therefore, to improve the reliability, error correcting codes (ECC) could be used at the cost of some additional hardware. Alternatively, we could use different challenges that have been evaluated to provide more reliable responses. Since there are multiple challenges that can produce the same response, we could apply several of these redundant challenges to improve the reliability and eliminate the ECC. Note that we have proposed a smaller subset of responses which doesn't reduce the entropy of the PUF as the attacker need to find the final *n*-bit key after the chip activation.

6. Evaluation

We have evaluated the effectiveness of our proposed technique by implementing it on an fiweld programmable gate arrays (FPGA) and calculating the hardware overhead. Note that in this paper, we do not report the overhead of the PUF implementation on the FPGA. The overhead of PUF implementations on FPGAs are well reported [\[24](#page-10-0)[–26\]](#page-10-1), and our approach does not depend on any specific strong PUF implementation. Furthermore, if a PUF is already used in a circuit for hardware security/authentication, the same PUF could be used for obfuscation key generation as well. We have performed the experiments on a number of ISCAS85-based obfuscation benchmarks [\[21\]](#page-9-17). Again, our approach will work with any key-based obfuscation method. For the purposes of these experiments, we have used logic cone size-based obfuscation [\[27\]](#page-10-2) along with the key generated from the strong PUF. In the logic cone size-based obfuscation method, the locked logic is placed in the largest logic cones so that it can impact more signals. A weighted normalized metric for all gates is measured and compared to find the position of the largest logic cone in the module under consideration. Both the fanin and fanout cones of a gate are considered as the metric. Therefore, gates with a higher value of this metric will have higher fanin and fanout and will be chosen as a locking gate in the inputs.

The benchmarks have been synthesized using Xilinx Vivado. Table [1](#page-7-0) shows the characteristics of the ISCAS85 benchmarks under evaluation and the FPGA utilization as reported by Vivado. We have illustrated comprehensive area overhead evaluations on the ISCAS85 benchmarks suite in Table [2.](#page-7-1) The benchmark circuit name is shown in the first column. The second column demonstrates the different key sizes used for obfuscation. Columns 3 and 4 show the area overhead in terms of LUTs due to obfuscation and the overhead that key generation adds to obfuscation after implementation on FPGA. Overheads in percentages have been shown in the last two columns. Figure [5](#page-7-2) shows the area overhead for the key generation compared to the base obfuscation method [\[27\]](#page-10-2) with different key sizes and benchmark circuits. It is noted that the area overhead for the key generation is typically less than 4% for different benchmarks and key sizes.

Circuit	Inputs	Outputs	LUTs	Slices
C880	60	26	66	25
C3540	50	22	249	67
C ₅₃₁₅	178	123	267	89
C7552	207	108	291	93

Table 1. ISCAS85 circuit characteristics.

Table 2. Area overhead for obfuscation with key generation for different benchmarks with varying key size.

Circuit	Key Size	Obfuscated Area (LUT)	Obfuscated Area with Key Gen (LUT)	Area OH $(\%)$	OH (%) for Key Gen	
C880	32	84	85	27.27	1.19	
	64	94	103	42.42	9.57	
	128	105	109	59.09	3.81	
	256	138	142	109.09	2.90	
C3540	32	255	256	2.41	0.39	
	64	257	261	3.21	1.56	
	128	309	316	24.10	2.27	
	256	343	356	37.75	3.79	
C ₅₃₁₅	32	290	296	8.61	2.07	
	64	307	311	14.98	1.30	
	128	342	351	28.09	2.63	
	256	452	457	69.09	1.11	
C7552	32	316	320	8.59	1.27	
	64	347	348	19.24	0.29	
	128	359	370	23.37	3.06	
	256	463	468	59.11	1.08	

Figure 5. Overheads for key generation compared with base obfuscation.

Depending on the internal design and architecture, a PUF can take multiple clock cycles to evaluate the response. In Table [3,](#page-8-2) we show the impact of the evaluation time on the overhead. As the clock cycles increase, the number of registers increase logarithmically primarily to keep timing state. We also see the difference in utilization for a 16-bit PUF $(n' = 16)$ compared to a 32-bit PUF $(n' = 32)$. Consider the 1-clock cycle 64-bit key case. For a 16-bit PUF, we need 67 registers total—64 registers for the key, two registers to keep track of the four sub-keys, and one more register for control logic state. As we increase *n'*, the key generation cost goes down slightly because *S* goes down. However, it comes at the expense of a larger PUF and more effort to find the challenges (Equation [\(4\)](#page-4-1)).

Key Size (PUF Delay)	4-Bit PUF		16-Bit PUF		32-Bit PUF	
	#LUTs	#Registers	#LUTs	#Registers	#LUTs	#Registers
32 bit (1 clock cycle)	3	36	$\mathbf{1}$	34	$\mathbf{1}$	33
64 bit (1 clock cycle)	$\overline{4}$	69	2	67	1	66
128 bit (1 clock cycle)	6	134	2	132	$\overline{2}$	131
256 bit (1 clock cycle)	7	263	4	261	2	260
32 bit (2 clock cycle)	4	37	2	35	2	34
64 bit (2 clock cycle)	5	70	3	68	3	67
128 bit (2 clock cycle)	7	135	3	133	3	132
256 bit (2 clock cycle)	8	264	5	262	3	261
32 bit (4 clock cycle)	4	38	2	36	$\overline{2}$	35
64 bit (4 clock cycle)	5	71	3	69	3	68
128 bit (4 clock cycle)	7	136	3	134	3	133
256 bit (4 clock cycle)	8	265	5	263	4	262
32 bit (8 clock cycle)	5	39	3	37	3	36
64 bit (8 clock cycle)	6	72	$\overline{4}$	70	4	69
128 bit (8 clock cycle)	8	137	$\overline{4}$	135	$\overline{4}$	134
256 bit (8 clock cycle)	9	266	6	264	4	263

Table 3. Area (lookup table (LUT)) and slice registers for different PUF bits for different clock cycles.

7. Conclusions

We have proposed a strong PUF-based obfuscation method so that the functionality of the logic is completely hidden. The approach is evaluated on a number of different ISCAS85 benchmark circuits. Compared to LUT-based obfuscation, our approach uses significantly less hardware overhead to provide obfuscation of much larger circuit blocks with similar security guarantees. Compared to key-based logic locking obfuscation, our PUF methodology provides several advantages for very little overhead. Since the key is never applied directly, it provides significant protection against key leakage. In addition, by using a PUF, the activation sequence for each chip is unique, limiting the chance that a key can be leaked. We have also presented a mathematical model to help estimate the effort required to find the sub-keys to make a final key for obfuscation. This effort calculation can be used to choose an appropriate *n'* and *S* value based on security needs. Overall, our proposed method is a viable solution against IC piracy/overproduction through the use of strong PUF-based key generation for obfuscation.

Author Contributions: Conceptualization, M.S.E.Q. and J.A.C.; Methodology, M.S.E.Q.; Software, M.S.E.Q.; Supervision, J.A.C.; Validation, M.S.E.Q. and J.A.C.; Formal analysis, M.S.E.Q. and J.A.C.; Visualization, J.A.C.; Data curation, M.S.E.Q.; Project administration, J.A.C.; Writing—original draft preparation, M.S.E.Q.; Writing—review and editing, M.S.E.Q. and J.A.C.; Project administration, J.A.C.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

References

- 1. Guin, U.; Huang, K.; DiMase, D.; Carulli, J.M.; Tehranipoor, M.; Makris, Y. Counterfeit integrated circuits: A rising threat in the global semiconductor supply chain. *Proc. IEEE* **2014**, *102*, 1207–1228. [\[CrossRef\]](http://dx.doi.org/10.1109/JPROC.2014.2332291)
- 2. Rostami, M.; Koushanfar, F.; Rajendran, J.; Karri, R. Hardware security: Threat models and metrics. In Proceedings of the International Conference on Computer-Aided Design, San Jose, CA, USA, 18–21 November 2013; IEEE Press: Piscataway, NJ, USA, 2013; pp. 819–823.
- 3. Quadir, S.E.; Chen, J.; Forte, D.; Asadizanjani, N.; Shahbazmohamadi, S.; Wang, L.; Chandy, J.; Tehranipoor, M. A survey on chip to system reverse engineering. *ACM J. Emerg. Technol. Comput. Syst. (JETC)* **2016**, *13*, 6. [\[CrossRef\]](http://dx.doi.org/10.1145/2755563)
- 4. Roy, J.A.; Koushanfar, F.; Markov, I.L. EPIC: Ending piracy of integrated circuits. In Proceedings of the Conference on Design, Automation and Test in Europe, Munich, Germany, 10–14 March 2008; ACM: New York, NY, USA, 2008; pp. 1069–1074.
- 5. Baumgarten, A.C. Preventing Integrated Circuit Piracy Using Reconfigurable Logic Barriers. Ph.D. Thesis, Iowa State University, Ames, IA, USA, 2009.
- 6. Rajendran, J.; Pino, Y.; Sinanoglu, O.; Karri, R. Security analysis of logic obfuscation. In Proceedings of the 49th Annual Design Automation Conference, San Francisco, CA, USA, 3–7 June 2012; ACM: New York, NY, USA, 2012; pp. 83–89.
- 7. Chakraborty, R.S.; Bhunia, S. HARPOON: An obfuscation-based SoC design methodology for hardware protection. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.* **2009**, *28*, 1493–1502. [\[CrossRef\]](http://dx.doi.org/10.1109/TCAD.2009.2028166)
- 8. Rajendran, J.; Sam, M.; Sinanoglu, O.; Karri, R. Security analysis of integrated circuit camouflaging. In Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security, Berlin, Germany, 4–8 November 2013; pp. 709–720.
- 9. Alkabani, Y.; Koushanfar, F. Active Hardware Metering for Intellectual Property Protection and Security. In Proceedings of the USENIX Security Symposium, Boston, MA, USA, 6–10 August 2007; pp. 291–306.
- 10. Zamanzadeh, S.; Jahanian, A. Higher security of ASIC fabrication process against reverse engineering attack using automatic netlist encryption methodology. *Microprocess. Microsyst.* **2016**, *42*, 1–9. [\[CrossRef\]](http://dx.doi.org/10.1016/j.micpro.2015.11.017)
- 11. Khaleghi, S.; Da Zhao, K.; Rao, W. IC piracy prevention via design withholding and entanglement. In Proceedings of the 20th Asia and South Pacific Design Automation Conference, Chiba, Japan, 19–22 January 2015; pp. 821–826.
- 12. Koushanfar, F. Hardware metering: A survey. In *Introduction to Hardware Security and Trust*; Springer: Berlin, Germany, 2012; pp. 103–122.
- 13. Baumgarten, A.; Tyagi, A.; Zambreno, J. Preventing IC piracy using reconfigurable logic barriers. *IEEE Des. Test Comput.* **2010**, *27*, 66–75. [\[CrossRef\]](http://dx.doi.org/10.1109/MDT.2010.24)
- 14. Khaleghi, S.; Rao, W. Hardware Obfuscation Using Strong PUFs. In Proceedings of the 2018 IEEE Computer Society Annual Symposium on VLSI (ISVLSI), Hong Kong, China, 8–11 July 2018; pp. 321–326.
- 15. Wendt, J.B.; Potkonjak, M. Hardware obfuscation using PUF-based logic. In Proceedings of the 2014 IEEE/ACM International Conference on Computer-Aided Design (ICCAD), San Jose, CA, USA, 2–6 November 2014; pp. 270–271.
- 16. Koushanfar, F. Provably secure active IC metering techniques for piracy avoidance and digital rights management. *IEEE Trans. Inf. Forensics Secur.* **2012**, *7*, 51–63. [\[CrossRef\]](http://dx.doi.org/10.1109/TIFS.2011.2163307)
- 17. Devadas, S.; Suh, E.; Paral, S.; Sowell, R.; Ziola, T.; Khandelwal, V. Design and implementation of PUF-based 'unclonable' RFID ICs for anti-counterfeiting and security applications. In Proceedings of the 2008 IEEE International Conference on RFID, Las Vegas, NV, USA, 16–17 April 2008; pp. 58–64.
- 18. Herder, C.; Yu, M.D.; Koushanfar, F.; Devadas, S. Physical unclonable functions and applications: A tutorial. *Proc. IEEE* **2014**, *102*, 1126–1141. [\[CrossRef\]](http://dx.doi.org/10.1109/JPROC.2014.2320516)
- 19. Quadir, S.E.; Chandy, J.A. Low Pass Filter PUF: Authentication of Printed Circuit Boards Based on Resistor and Capacitor Variations. *Int. J. High Speed Electron. Syst.* **2018**, *27*, 1840021. [\[CrossRef\]](http://dx.doi.org/10.1142/S0129156418400219)
- 20. Alkabani, Y.; Koushanfar, F.; Potkonjak, M. Remote activation of ICs for piracy prevention and digital right management. In Proceedings of the 2007 IEEE/ACM International Conference on Computer-Aided Design, San Jose, CA, USA, 5–8 November 2007; pp. 674–677.
- 21. Amir, S.; Shakya, B.; Xu, X.; Jin, Y.; Bhunia, S.; Tehranipoor, M.; Forte, D. Development and Evaluation of Hardware Obfuscation Benchmarks. *J. Hardw. Syst. Secur.* **2018**, *2*, 142–161. [\[CrossRef\]](http://dx.doi.org/10.1007/s41635-018-0036-3)
- 22. Rührmair, U.; Sehnke, F.; Sölter, J.; Dror, G.; Devadas, S.; Schmidhuber, J. Modeling attacks on physical unclonable functions. In Proceedings of the 17th ACM Conference on Computer and Communications Security, Chicago, IL, USA, 4–8 October 2010; pp. 237–249.
- 23. Becker, G.T. The gap between promise and reality: On the insecurity of XOR arbiter PUFs. In Proceedings of the International Workshop on Cryptographic Hardware and Embedded Systems, Saint Malo, France, 13–16 September 2015; pp. 535–555.
- 24. Yu, M.D.; Devadas, S. Secure and robust error correction for physical unclonable functions. *IEEE Des. Test Comput.* **2010**, *27*, 48–65. [\[CrossRef\]](http://dx.doi.org/10.1109/MDT.2010.25)
- 25. Majzoobi, M.; Koushanfar, F.; Potkonjak, M. Techniques for design and implementation of secure reconfigurable PUFs. *ACM Trans. Reconfigurable Technol. Syst. (TRETS)* **2009**, *2*, 5. [\[CrossRef\]](http://dx.doi.org/10.1145/1502781.1502786)
- 26. Meguerdichian, S.; Potkonjak, M. Device aging-based physically unclonable functions. In Proceedings of the 2011 48th ACM/EDAC/IEEE Design Automation Conference (DAC), New York, NY, USA, 5–9 June 2011; pp. 288–289.
- 27. Narasimhan, S.; Chakraborty, R.S.; Chakraborty, S. Hardware IP protection during evaluation using embedded sequential trojan. *IEEE Des. Test Comput.* **2012**, *29*, 70–79. [\[CrossRef\]](http://dx.doi.org/10.1109/MDT.2012.2205997)

 c 2019 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license [\(http://creativecommons.org/licenses/by/4.0/\)](http://creativecommons.org/licenses/by/4.0/.).