Efficient design of Oscillator based Physical Unclonable Functions on Flash FPGAs

Ugo Mureddu, Oto Petura, Nathalie Bochard, Lilian Bossuet, Viktor Fischer
Univ Lyon, UJM-Saint-Etienne, CNRS, Laboratoire Hubert Curien UMR 5516,
F-42023, SAINT-ETIENNE, France
Email: (ugo.mureddu,oto.petura,nathalie.bochard,lilian.bossuet,fischer)@univ-st-etienne.fr

Abstract—**With the scaling down of electronic devices and the boom of wireless communications, more and more smart devices are interconnected in what we call the Internet of Things. Connecting devices of everyday use can greatly improve our comfort, but it can also introduce unprecedented security problems. With billions of devices connected there is a huge risk of unauthorized use. In this context, Physical Unclonable Functions (PUFs) are a promising solution since they extract device intrinsic fingerprint that can be used for hardware identification and authentication. Here we present the first fully functional implementation of Oscillator based PUFs on Flash based FPGA. The implementation is presented for the Ring Oscillator based PUF and the Transient Effect Ring Oscillatory based PUF. After explaining those two PUF principles, we give all the necessary design practices to follow to obtain an efficient PUF implementation on Flash FPGA. Finally, we present the characterization of the PUFs and compare it to previous work. To the best of our knowledge, it is the first work which deals with the implementation of Oscillator based PUF on Flash FPGAs. Moreover, all design files are available online to ensure repeatability.**

1. Introduction

We live in a smart world. The Internet is widely available, connection costs are low and many devices are emerging with Wi-Fi capabilities and sensors built into them. This impacts how we live but also how we work. From smart phones through smart homes and industry up to smart and self-driving cars everything is connected. All of these things are creating what we call the Internet of Things (IoT). This is the concept of connecting any device with an on/off switch to each other through the Internet without the need for human interaction.

Connecting the appliances of everyday use can greatly improve our comfort, but it can also introduce unprecedented security problems. When deployment and integration of IoT is growing, one of the main challenges is to provide security solutions regarding privacy and trust issues. With billions of devices connected there is a huge risk of unauthorized use or abuse of these devices. To protect from such risks we need security mechanisms allowing for a per-device authentication and authorization, which are built-in from the very first design stages.

In this context, Physical Unclonable Functions (PUFs) are a promising solution since they provide device intrinsic fingerprint usable for secure hardware identification and authentication.

Moreover, since any kind of device can be a part of the IoT, we can face two constraints: cost and size. Most of the IoT devices are embedded in existing systems, thus there is not much space to fit in. Also since these devices are deployed in thousands or millions, the cost is extremely important.

Within these constraints, using a Flash memory based FPGAs is advantageous for multiple reasons. Firstly, we can reduce the overall materials cost because there is no need for a configuration memory.

The next advantage for IoT applications is lower demand on power supply. The Flash FPGA does not need to be configured at boot, so there is no configuration power peak. The whole design consumes only the power it needs to work. Thus there is no need to over estimate the power supply needs just to compensate for configuration power consumption, which leads to lower power supply costs as well as lower demands on heat sinks and physical space.

Last, but not least, the Flash FPGAs can be safely deployed in industrial environments where there might be a lot of electromagnetic noise, which might disturb volatile SRAM memories. Since industrial automation is one of the main IoT applications, this aspect is also non negligible.

Contribution: Our main contribution is the first proposal of design methodologies leading to a fully functional Oscillator PUF on Flash memory based FPGAs. This work is aiming to enlarge the scope of PUF implementations on FPGAs when all previous works are on SRAM FPGAs [1], [2], [3], [4]. Implementing an efficient PUF on Flash FPGAs is not an easy task.

Structure: In the next section we provide all the necessary background informations on PUF. Section 3 describes the Ring Oscillator (RO) and Transient Effect Ring Oscillator (TERO) PUF systems implemented and all the design practices leading to an efficient implementation. In section 4 we present the characterization metrics and compare results of both PUFs to non-optimized designs. Finally, we conclude in section 5.

2. Physical Unclonable Functions

PUFs are hardware primitives that use manufacturing process variations (MPVs), such as the mismatch between transistors, to generate a device-specific output, which usually is a binary number. This output can be seen as the fingerprint of a device. It is called a response.

Basic applications include identification [5], authentication [6] and key generation [7].

Many PUF principles have been published up to now. The most known are memory based PUFs including for example SRAM PUFs [8] and delay based PUFs such as arbiter PUFs [9], ring oscillator PUFs [10], RS latch PUFs [11] and transient effect ring oscillator PUFs [1]. Although some FPGAs, including Flash FPGAs (*i.e.* Microsemi SmartFusion 2), are provided with SRAM based PUF hardware IPs, Helfmeier et al. have shown that it is easily clonable [12]. That's why another PUF implementation for FPGA must be found. Moreover, studies have shown that PUF principles using oscillating circuitries are the best candidates for FPGAs [2], [3]. For this reason, RO PUF and TERO PUF were selected.

3. Design and implementation

In this section, the PUF systems implemented and the design rules to follow ending up with efficient implementations on Flash FPGAs will be described. All implementations have been made on Microsemi SmarFusion2 FPGAs using Microsemi Libero design software v11.7. Design files, including constraint files are provided for free to ensure repeatability[1].

3.1. Oscillator PUF architecture

The implemented PUFs are composed of 256 oscillating cells, two counters and a bit extractor as depicted in Figure 3.a. Cells are divided in two blocks, A and B. To avoid correlation, a cell of the block A is always compared to a cell of the block B and is used only once. One cell per block is selected using two demultiplexers and two multiplexers are placed right after the cell blocks in order to drive the correct cell outputs to the clock counters. The cell selection is usually called a challenge.

ROs are digital oscillators consisting of an odd number of inverters connected to form a loop. Figure 1 shows a typical RO cell.

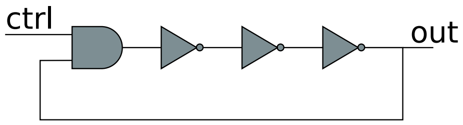

Figure 1. RO cell

1. https://gitlab.univ-st-etienne.fr/ugo.mureddu/flash_fpga_puf_source_code.git

An RO-PUF extracts MPVs in a digital circuit by comparing the oscillation frequencies of two identically implemented ROs. Compared RO cells are triggered at the same time. As soon as one of the counters reaches a maximum value, an arbiter stops them. If it is the one from the block A, resp. block B, the arbiter generates a '1', resp. '0'. This is done for all implemented ROs and the resulting bits are concatenated to form the 128-bits response.

TEROs corresponds to a very specific configuration of an RS latch, with two inputs featuring the same voltage and additional gates, that oscillate temporarily [13]. Figure 2 shows a typical TERO cell.

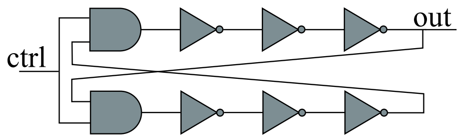

Figure 2. TERO cell

Here, we compare the number of oscillations. That's why the bit extractor is a subtractor. With this structure we can extract from 1 to 3 bits per challenge. As explained in [4], counters and activation time of the control signal need to be sized accordingly to the mean number of oscillations of the TERO cells. For this work, counters are 11 bits and activation time is set to $1\mu s$. Since it is not the main subject of this article this dimensioning will not be further detailed.

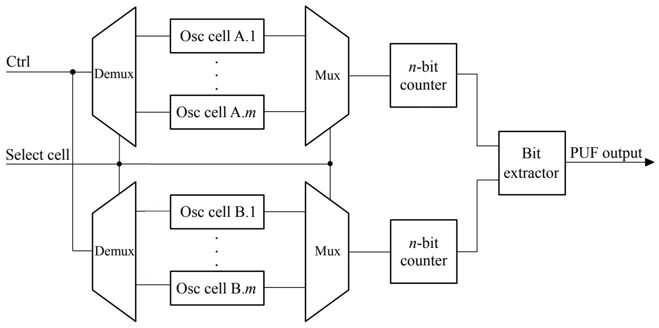

Figure 3. Classic oscillator PUF architecture

3.2. Proposed PUF design methodologies

3.2.1. General design. If the implementation is not done carefully, it can happen that the comparison of two cells depends on their position on the FPGA. If this is the case, some comparisons tend to be biased towards a certain value. For example if an oscillating cell of the block A is routed with a bigger delay, its frequency will be lower and the comparison with a cell of the block B will be biased towards '0' on all chips. That's why, in order to get frequency variations due to MPVs and not due to differences in routing, it is really important to have cells identically implemented.

Unaltered cell structure. For that, the first thing to do is to keep the cell structure unaltered during all the synthesis, place and route processes of the Libero design software. Indeed, Libero is always optimizing the code and unfortunately, there is no way to prevent the optimization. Still there are few things designers can do. First, disable the retiming optimization in synthesis options because it will put loop breakers in the cells and thus modify its structure. Then, the VHDL code needs to be as low level as possible to be sure that when it is synthesized it is not modified (*i.e.* number of inverting elements). To do so, it is necessary to only use the components of the SmartFusion2 library. This library provides basic element of a 2-inputs AND gate (*AND2*) and an inverter gate (*INV*) which will not be optimized by the tool.

Delay control. Delays between gates in an FPGA can be really important since by definition an FPGA is composed of an array of gates. If it is necessary to cross many gates (or buffers) to connect two elements, it is hard to control the delay. That's why, elements of the cell have to be rigorously placed side by side. This is achieved using the *Chip Planner Constraint Manager* or a *pdc constraint file* where elements can be precisely placed in LUTs.

Exclusive regions. To ensure that no routing can cross oscillating cells, exclusive regions need to be created. They are physical regions on the FPGA where only assigned elements can be placed. This way during the synthesis, place and route process nothing else will be added inside. To also avoid routing to cross those regions, the designer has to select the *constrain routing* option when creating the region. For the PUF implementation, it is necessary to create one region per cell block.

Maximizing local mismatches. Assuming that, at this point, cells are identically implemented, the frequency difference between them is only due to MPVs. However, two sources of mismatch affect FPGAs. The first one is called local variations (variations at transistor level) and the second one global variations (variations at chip level). Global variations involve a gradient over the chip. To maximize the randomness of the PUF response the mismatch between transistors have to be free of any gradient. Otherwise, a bias can occur. Thus, the two oscillating cell regions have to be as close as possible.

3.2.2. RO PUF restrictions. The delay path to control for the RO PUF is much longer than the one of the TERO PUF. Indeed, to get a fair comparison, cells have to be triggered at the same time and the delays between output cells and input counters need to be the same. Thus, the two counters are placed side by side and multiplexer outputs are sent to clock buffers (*CLKBUF*) which route signal to the global clock network. The global network is composed of global buffers to distribute low-skew clock signals. This way the delay between outputs of ROs and inputs of counters is better controlled. It is not possible to do the same with the enable signal because global resources are limited so a flip-flop is placed just before each input of the RO cells. Since the clock signal is distributed with low skew, flip-flops will switch at the same time.

Moreover, the output signal of an RO cell has poor characteristics (*i.e.* distorted rising and falling edges or asymmetric duty cycle). To enhance the signal quality a T flip-flop is placed at the output of each RO cell. This acts as a buffer and improves the quality of the edges. Moreover, the duty cycle at the output of the flip-flop is symmetric. Finally, it divides the oscillation frequency by two which reduces counter failure possibilities without affecting the mismatch between cells. This must not be done for TERO cells because they are very sensitive to output load [13]. To put a T flip-flop at the output would unbalance the two branches and so drastically decrease the number of oscillations.

Finally, the oscillations have to be counted for a certain time. Otherwise, the comparison is noisy. The more you count, the more you accumulate transistor's mismatch and the less the noise impacts results. Indeed the mean value of the noise as a function of time is null when the transistor mismatch is increasing with time. Thus, the size of the counters have to be dimensioned according to the design. This will improve the stability of the responses.

4. PUFs characterization

4.1. Characterization metrics

The three commonly used metrics to characterize a PUF are uniqueness, steadiness and randomness [14]. Indeed, a good PUF must generate responses on different devices that are unique. Moreover, responses to the same challenge on the same device should be stable over time and operating conditions. Finally, the outcome of a PUF must not be predictable. Thus, the output values '1' and '0' should be distributed equally and no correlation between bits should occur.

Uniqueness. It shows to what extent responses generated by the PUF on different devices are unique. Considering two chips i and j, and responses $R_{i,y}$ and $R_{j,y}$ respectively, the average $Uniqueness$ for a set of n chips generating x responses is defined in Equation 1:

$$Uniqueness = \frac{1}{n(n-1)x} \sum_{i=1}^{n} \sum_{\substack{j=1 \\ j \neq i}}^{n} \sum_{y=1}^{x} \frac{HD(R_{i,y}, \overline{R_j})}{n} \tag{1}$$

where $\overline{R_j}$ is the reference response of j coming from average of samples $R_{j,y}$ and HD the Hamming Distance between two responses.

With the right amount of data a perfect $uniqueness$ tend to a binomial distribution of parameters n the number of experiments and probability $p = 0.5$ where its $mean = np = 50\%$ and $variance = np(1-p) = 25\%$. If the $mean$ is different than 50%, it can either be that correlation between responses of different chips occurred or some responses are biased towards a certain value. Without the sufficient amount of data an acceptable $uniqueness$ can have a $variance$ lower than 25%. However, if the $variance$ is bigger than 25%, it indicates some cells correlation within chips. That's why it is important to plot the $uniqueness$ distribution and not only the $mean$ value.

Steadiness. Expresses how efficient is a PUF in reproducing a response to the same challenge on the same device. The *steadiness* defined in Equation 2 quantifies changes at the output of a PUF over many measurements.

$$Steadiness = \frac{1}{x}\sum_{y=1}^{x}\frac{HD(R_{i,y},\overline{R_i})}{n} \tag{2}$$

where $\overline{R_i}$ is reference response of i coming from average of samples $R_{i,y}$ at nominal conditions. A perfectly stable PUF implementation has a $Steadiness$ of 0%.

Randomness. This parameter is the most controversial since there is at the moment no ideal way to estimate the randomness of a PUF. Most of the time, it is done by measuring the bias (or bit-aliasing).

Pehl et al. [15] tried to establish some further evaluation approaches like joint entropy to identify bits correlation and therefore predictability weaknesses. Others proposed to use test suites like it is done for True Random Number Generators with NIST tests or AIS20/31 but to perform an acceptable entropy estimation the number of responses needed is very high. Thus millions of test chips are necessary, which does not seem possible.

We believe that one way would be to establish a stochastic model of the PUF prior to implementing it, which will be of course technology-dependent. This has not been done yet and we raise this question for future work.

Moreover, it has been proven by Feiten et al. [16] that uniqueness and bit-aliasing are mutually redundant. That's why as part of our study, uniqueness will act as bias estimation.

In this work, to minimize bits-correlation possibilities one cell of the block A is always compared to a cell of the block B which avoids them to influence each other. Moreover a single cell is only used for one bits generation.

Again, if those tests show some bias or correlation we can conclude the PUF has weaknesses but if not we can not affirm the PUF has full entropy.

4.2. Experimental setup

The characterization is performed on 24 Microsemi SmartFusion2 FPGAs. In order to get a bigger amount of data, designs were locked and moved to different part of the FPGAs. This way the same PUF implementation extracts other MPVs because it is not using the same transistors. Thus characterization is made of 48 PUF implementations. Each 128-bits response was generated one thousand times.

4.3. Characterization results

Uniqueness. Figures 5 and 6 depict the *uniqueness* histograms of the PUFs for a non-optimized implementation as well as for an implementation following the proposed design methodologies (see section 3.2). The fitted normal approximation is also plotted. Moreover, the normal approximation of an ideal *uniqueness* ($\mathcal{N}(\mu = 50\%, \sigma^2 = 25\%)$) is drawn in dashed line where μ is the mean and σ^2 is the variance. Figure 5 shows the results of the RO PUF implementation and Figure 6 the TERO PUF results.

Improvements are clearly visible since the *mean uniqueness* of the RO PUF, respectively TERO PUF, goes from 12.1% to 42.2%, respectively 28.3% to 48.1%, with the proposed design methodologies. $Variance$ is also in appropriate ranges since it is below 25% for both implementations.

However results of the RO PUF still disclose either a correlation between responses of different implementations or a bias towards 0. To find out, it is necessary to check if some bits of the responses are always the same on all the PUF implementations. It turns out that there is at least one bit of the response which is always 0. This explains the offset of the *uniqueness* distribution towards 0.

If after improvements, routing differences are still more important than MPVs it can mean that either the RO-PUF is not extracting MPVs well enough on Flash FPGAs or the design still needs some optimizations. One thing that could be further explored is the delay between ROs and output of the multiplexers. To better control this delay the multiplexers could also be made with components of the SmartFusion2 library since there is a *MUX4* element.

Steadiness. After following the proposed design methodologies, the *mean steadiness* is 1.68% for the RO PUF and 1.52% for the TERO PUF. Before, it was 0.69% for the RO PUF and 1.3% for the TERO PUF.

However, when the *uniqueness* is too far from 50%, the *steadiness*, even close to 0%, is no longer representative because the PUF has a very strong bias. Looking at a single statistical parameter cannot lead to the conclusion that PUF quality is sufficient. That's why it is important to observe both, *uniqueness* and *steadiness*.

Moreover, *steadiness* of the proposed RO PUF over voltage variations have been studied for different counter sizes. Figure 4 depicts the result where $1.2V$ is the nominal voltage of the device.

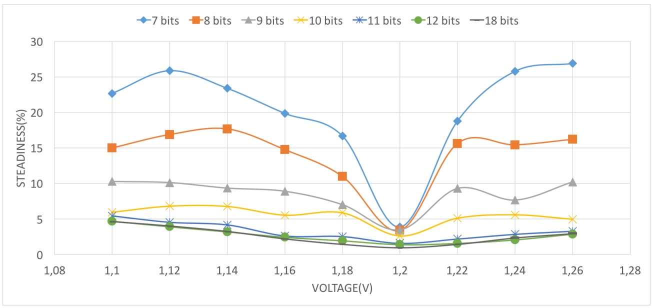

Figure 4. *Steadiness* of the RO-PUF over voltage variations

It is very distinct, especially at voltage corners, how the counter size affects the stability. For a 7-bits counter, the stability can vary up to 27% when for an 11-bits counter, it is not going more than 5%. However after a certain counter size, improvements are no longer evolving. That's why in this case it is not needed to use a counter bigger than 11-bits.

TABLE 1. Summary of characterization results of the RO and TERO PUFs

PUF metrics	**RO-PUF**			**TERO-PUF**			
Hardware Target	Xilinx [2] Spartan-3E	Microsemi SmartFusion 2		Xilinx [4] Spartan-6	Altera [4] Cyclone-V	Microsemi SmartFusion 2	
		Non-optimized	Optimized			Non-optimized	Optimized
Uniqueness	47.3%	12.1%	42.2%	48.5%	47.6%	28.3%	48.1%
Steadiness	0.9%	not interpretable	1.68%	2.6%	1.8%	not interpretable	1.52%

This is not a general truth. Dimensioning of the counters needs to be done for each implementation.

Comparison. In order to compare a non optimized design with an optimized design, results are summarized in Table 1. It depicts $mean\ uniqueness$ and $mean\ steadiness$ for each of them. Results from implementations of RO-PUFs on Xilinx Spartan 3 [2] and TERO-PUF on Xilinx Spartan 6 [4], Altera Cyclone V [4] are also depicted. From what can be observed the proposed methodologies increase plainly the PUF quality. As explained before, $steadiness$ can not be interpreted if the uniqueness is too far from 50%. That's why, for the non-optimized design $steadiness$ is reported as not interpretable in the table. The TERO implementation is more efficient than the RO-PUF since it has a better $uniqueness$. This is probably due to the fact that the delay path to control is shorter which makes it easier to implement. Indeed, we can see that even with a sufficient $steadiness$ the RO PUF has a lack in terms of statistical quality since there is a slight correlation between responses of different implementations. Compared to other implementations the optimized design of the TERO PUF is in appropriate range with a 1.52% $mean\ steadiness$ and 48.1% $mean\ uniqueness$.

5. Conclusion

In this paper, efficient designs of an RO PUF and a TERO PUF have been proposed for the first time on Flash based FPGAs. Those implementations have been made at the lowest possible level. It shows a number of improvements: First, how to keep the cell structure unaltered. Second, how to control the delay of the critical elements of the PUF. Third, a way to isolate oscillating cells from disturbing elements. Fourth, the maximization of the local mismatches. And finally, how to improve signal quality at the output of the RO cells. Characterization results have shown the efficiency gain with an optimized design.

Acknowledgments

This work has received funding from the European Unions Horizon 2020 research and innovation programme in the framework of the project HECTOR (Hardware Enabled Crypto and Randomness) under grant agreement No 644052.

The authors are also grateful to Serigne Sarr for his characterization of the RO-PUF reliability according to the number of bits counter.

References

[1] L. Bossuet, X. T. Ngo, Z. Cherif, and V. Fischer, "A PUF based on a transient effect ring oscillator and insensitive to locking phenomenon," *IEEE Trans. Emerging Topics Comput.*, vol. 2, no. 1, pp. 30–36, 2014.

[2] A. Maiti, J. Casarona, L. McHale, and P. Schaumont, "A large scale characterization of RO-PUF," in *Proceedings of the 2010 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, Anaheim Convention Center, California, USA, 13-14 June 2010, pp. 94–99.

[3] S. Morozov, A. Maiti, and P. Schaumont, *An Analysis of Delay Based PUF Implementations on FPGA*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 382–387.

[4] C. Marchand, L. Bossuet, and A. Cherkaoui, "Design and characterization of the TERO-PUF on SRAM FPGAs," in *2016 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, July 2016, pp. 134–139.

[5] R. Maes, P. Tuyls, and I. Verbauwhede, "Statistical analysis of silicon PUF responses for device identification," in *Workshop on Secure Component and System Identification-SECSI*, 2008.

[6] G. Hammouri, E. Öztürk, B. Birand, and B. Sunar, *Unclonable Lightweight Authentication Scheme*. Springer Berlin Heidelberg, 2008, pp. 33–48.

[7] B. Škorić, P. Tuyls, and W. Ophey, *Robust Key Extraction from Physical Uncloneable Functions*. Springer Berlin Heidelberg, 2005, pp. 407–422.

[8] Y. Su, J. Holleman, and B. Otis, "A digital 1.6 pj/bit chip identification circuit using process variations," *IEEE Journal of Solid-State Circuits*, vol. 43, no. 1, pp. 69–77, Jan 2008.

[9] G. E. Suh and S. Devadas, "Physical unclonable functions for device authentication and secret key generation," in *Proceedings of the 44th Design Automation Conference, DAC*, San Diego, CA, USA, June 4-8 2007, pp. 9–14.

[10] B. Gassend, D. Clarke, M. van Dijk, and S. Devadas, "Silicon physical random functions," in *Proceedings of the 9th ACM Conference on Computer and Communications Security*, ser. CCS '02. New York, NY, USA: ACM, 2002, pp. 148–160.

[11] B. Habib, J.-P. Kaps, and K. Gaj, *Efficient SR-Latch PUF*. Cham: Springer International Publishing, 2015, pp. 205–216.

[12] C. Helfmeier, C. Boit, D. Nedospasov, and J.-P. Seifert, "Cloning physically unclonable functions," in *IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, June 2013, pp. 1–6.

[13] L. M. Reyneri, D. D. Corso, and B. Sacco, "Oscillatory metastability in homogeneous and inhomogeneous flip-flops," *IEEE Journal of Solid-State Circuits*, vol. 25, no. 1, pp. 254–264, Feb 1990.

[14] A. Maiti, V. Gunreddy, and P. Schaumont, *A Systematic Method to Evaluate and Compare the Performance of Physical Unclonable Functions*. New York, NY: Springer New York, 2013, pp. 245–267.

[15] M. Pehl, M. Hiller, and H. Graeb, "Efficient evaluation of physical unclonable functions using entropy measures," *Journal of Circuits, Systems and Computers*, vol. 25, no. 01, p. 1640001, 2016.

[16] B. B. Linus Feiten, Matthias Sauer, "On metrics to quantify the inter-device uniqueness of PUFs," Cryptology ePrint Archive, Report 2016/320, 2016.

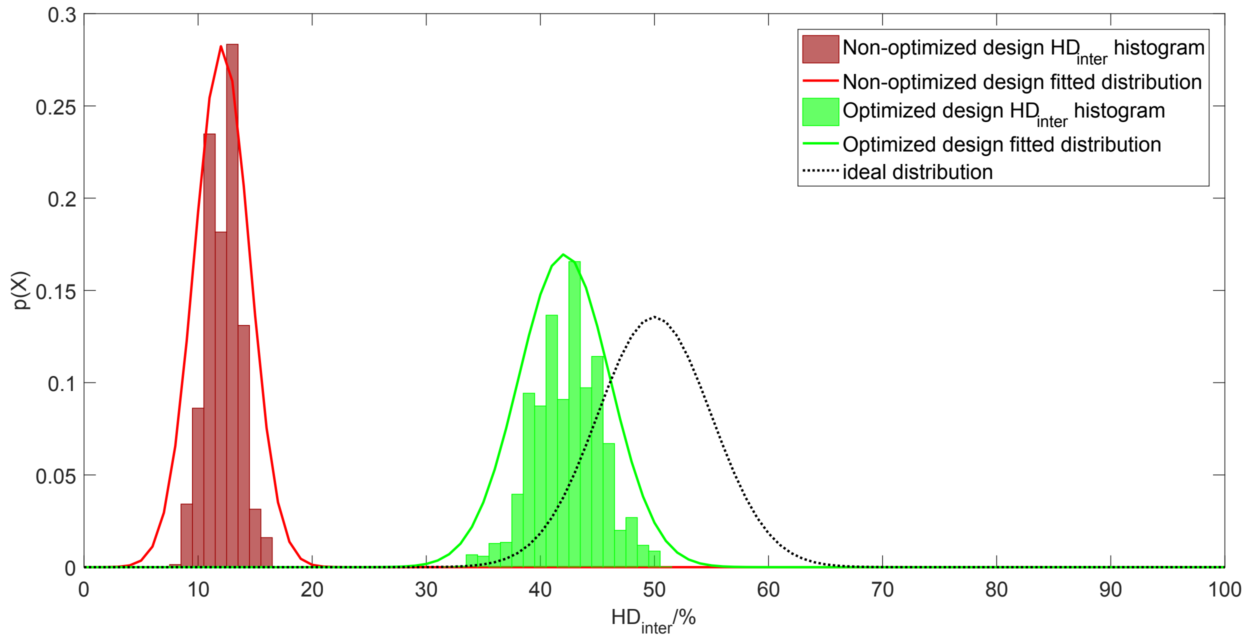

Figure 5. RO-PUF *uniqueness*

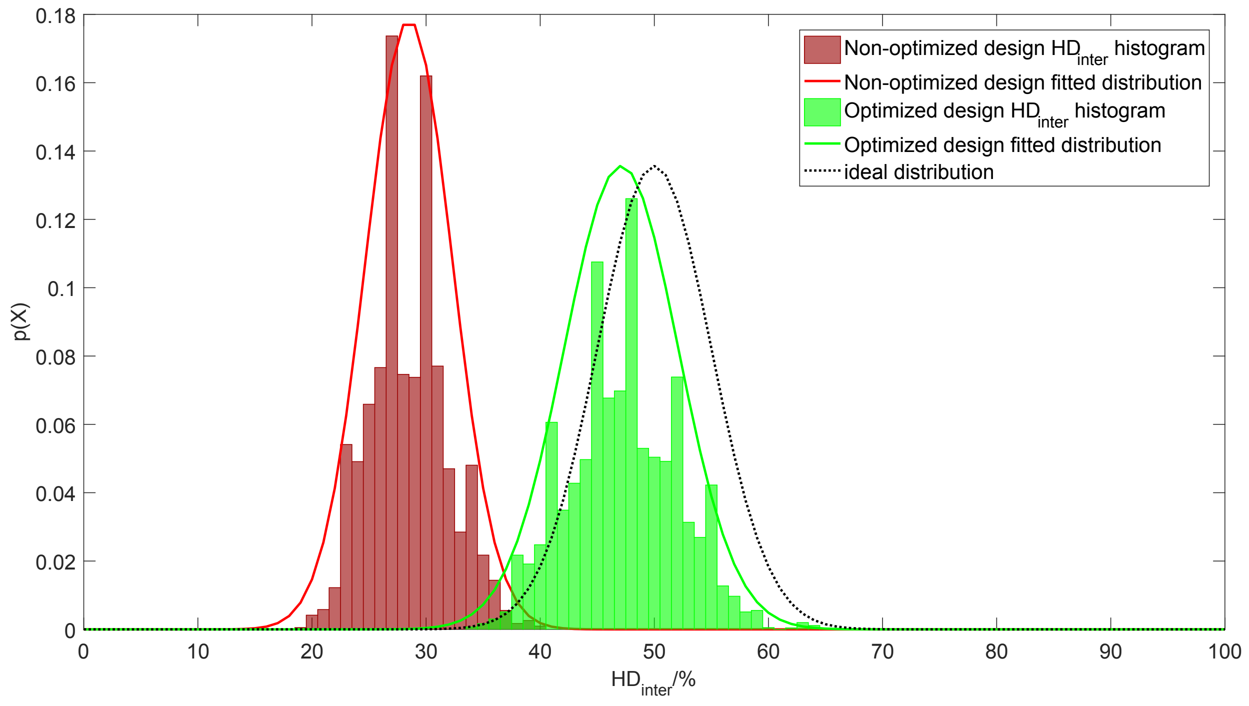

Figure 6. TERO-PUF *uniqueness*