VNU Journal of Science: Comp. Science & Com. Eng, Vol. 39, No. 1 (2023) 68-78

VNU Journal of Science:
Computer Science and Communication Engineering

Journal homepage: http://www.jcsce.vnu.edu.vn/index.php/jcsce

Original Article

Stabilizing Techniques for Secure On-chip Key Generation Based on RO PUF

Van-Toan Tran, Quang-Kien Trinh, Van-Phuc Hoang*

Le Quy Don Technical University, 236 Hoang Quoc Viet, Hanoi, Vietnam

Received 14 November 2021
Revised 25 May 2022; Accepted 02 June 2022

Abstract: Due to intrinsic fluctuation of physical characteristics, response bit-strings of Physically Unclonable Functions (PUF) of the devices are not suitable to be used directly as seeds for random number generation or secure keys for data encryption. In order to take advantages of Ring Oscillator PUF (RO PUF) in hardware security applications, in this work, we propose several techniques and algorithms to stabilize the response data of a RO PUF scheme to provide a stable and unique output bit-string. These processes are dynamically evaluated on repeatedly samples the RO frequency values which do not require additional hardware as well as external physical memory and therefore, the security of the communication systems can be improved. The proposed methods are verified by experiments conducted on Xilinx Artix-7 FPGA devices.

Keywords: RO PUF, FPGA, hardware security, key generator.

1. Introduction

Physically Unclonable Functions (PUF) have been developed by Suh and Devadah in [1] based on the fundamental concept about the Physical One-way Function (POF) that can be considered as the IC fingerprint of the device. This promises vast area of security applications such as device identification and authentication, intellectual properties protection, secure key generation, and so on. Several works have been published to construct the theoretical model [2-4], proposed particular schemes [5, 6], and extend the area of applications [7-9]. The common working principle of different PUF schemes is utilizing the device mismatches to derive intrinsic properties. From the implementation aspect, the FPGA-based Ring Oscillator PUF (RO PUF) is one of the most widely used PUF schemes due to its layout's consistence against global variations and

* Corresponding author.
E-mail address: phuchv@lqdtu.edu.vn

https://doi.org/10.25073/2588-1086/vnucsce.306

operating condition [10], as well as exploit the advantage of FPGA technology on semi-custom hardware designs [11, 12].

From application aspect, a PUF-based key generator may satisfy the following requirements [13]. First of all, PUFs could offer a totally true random source due to inherent manufacturing process variations. Then, since PUF instance responses are device-primitives, the generated keys have a significant level of uniqueness. Finally, since keys are generated each time the PUF scheme is evaluated, no external memory is required to store the keys. The latter lowers the chances of being intervened, particularly the side-channel attacks.

Many works have been published in the field of PUF-based key generation that lay the theoretical framework together with introduce the key extraction schemes. In [1], the authors suggested a two-part scheme of initialization and re-generation, in which procedures of error correction encode and decode preserve the stable PUF output though under in unstable operating conditions. The authors in [9] established a novel variant of the RO PUF that generates keys via an entropy accumulator procedure. Therein, the primary components of the helper data scheme are Lehmer-Gray order encoding and a resource-optimized BCH decoder [14, 15]. By replacing the Hash function output with the syndrome from the BCH code, the authors in [16] enhanced the capability of the fuzzy extractor, resulting in the Hamming distance between keys that adapts with key size. The majority of the above methods have been suggested for the traditional RO PUF that obtains the bit-string by means of the sign function [1], which is extremely complicated and ineffective due to the intensive hardware resources consumption.

In [10, 17], we have introduced an ID extraction and authentication scheme using RO PUF and Euclidean metrics. The design is based on Suh and Devadas's conventional RO PUF [1], where nominal ID of the device has the form of a vector with coordinates are differential frequencies (dfs) of two successive identical in physical layout ROs, $df_i = f_i - f_{i+1}$, $i = \overline{1, n-1}$, where n is the number of ROs in an RO array. In the form of bit-string, the extracted ID is the combination of the binary presentation of differential frequencies. This method retains only the local variation while effectively removes the global variations impacts. As a result, the extracted IDs are distinguishable and steady with global factors such as ambient temperature, operating voltages, and so on. This also suggests using ID samples from ID extraction based on RO PUF scheme to coin stable and unique bit-string that can served as a seed for key generation. However, the instantaneous measured ID, on the other hand, shows a small fluctuation, which is caused by the unstable RO frequencies. Even though the df sample values are tightly distributed around the mean df value, their bit-string presentations are extremely fluctuated, particularly at least significant bits. Otherwise, we can not use the stable nominal ID from the ID database since this method may create a security leak. Figure 1 depicts the overall key extraction procedure. The difference frequencies are derived on-chip from absolute RO frequencies. A stabilizer is required to keep the output bit-string stable and unique. Finally, from the resulted bit-strings, a Hash function is used to produce the final random and secure key. This paper focuses on algorithms and techniques for stabilizing the output bit-string of an RO PUF scheme.

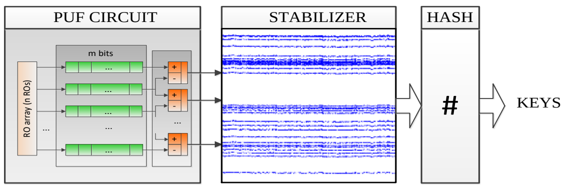


Figure 1. Key generation from PUF data and Hash function.

In [18], we have developed some techniques and methods to extract stable and unique bit-strings from RO PUF design. However, there is still an issue of hardware implementation to examine the efficient of on-chip bit-string extraction. Therefore, in this paper, we present more particular approach that consist of additional algorithms and details of implementation of the designs on FPGA. 18 The rest of the paper is structured as follows. After providing a brief introduction of RO PUF procedures and key generation using RO PUF designs in Section 1, we discuss algorithms and additional techniques to extracting stable and unique bit-strings as well as solutions to improve their efficiency in Section 2. Next, we present some detail implemented schemes respected to proposed methods in Section 3. Finally, Section 4 summarizes the paper.

2. Extracting Stable Keys and Additional Techniques

In this section, we introduce multiple methods and examine their efficiency in stabilizing the df samples in order to extract the unique bit-string. The input data was obtained by evaluating 5 different devices of Xilinx Artix-7 XC7A35T at the room temperature. The physical layout of the design is kept highly symmetric by regularly arranging the ROs within the same clock region. The absolute frequency of the RO yielded from count number of the system clock in a time interval with high accuracy. In [17], we have shown that the maximum absolute error in determining the RO frequency is inversely proportional to the time interval measurement (50 Hz corresponding to the period of 20 ms in our experiments). The relative error and accuracy of the measurement are 0.002% and 99.998%, respectively.

In this work, we propose the main approaches to solve the above problem as follows:

- Extracting the stable bit-string by cutting off the fixed number of variation bits in df sample value,

- Extracting the stable bit-string by using adaptive data mask,

- Extracting the most repeated element from statistical distribution, and an additional technique to improve the proposed method's efficiency, that is sample averaging technique as presented in the next subsection.

2.1. Stabilizing Bit-string by Averaging Method

Since of the high randomness in RO df values, it is not practical to use the df sample values directly, especially for the case that there are some outliers. In contrast, we use the averaging values of differential frequencies, that are extracted from number of df samples as follows.

Assume nominal differential RO frequency of RO_j is df_{j0}, $j = \overline{1, n_{ring} - 1}$, where n_{ring} is a number of ring oscillators. The differential frequency RO_j of the i – sample is:

$$df_{ij} = df_{j0} + \delta_{ij} \tag{1}$$

where δ_{ij} is the discrepancy of differential frequency of the respective sample. So, the average differential RO_j frequency is:

$$\begin{aligned} df_{mean,j} &= \frac{1}{n_{sample}} \sum_{i=1}^{n_{sample}} df_{ij} \\ &= df_{j0} + \frac{1}{n_{sample}} \sum_{i=1}^{n_{sample}} \delta_{ij} \end{aligned} \tag{2}$$

where n_{sample} is the number of samples.

The maximum of $\left| \frac{1}{n_{sample}} \sum_{i=1}^{n_{sample}} \delta_{ij} \right|$ for all RO pairs determines the number of excluded bits to obtain the unique key sample.

In order to prevent the overflow, the algorithm is modified as presented in Table 1. The number of sample essentially is chosen to be a power of 2 (e.g. 1024, 2048), hence, the division is simplified to be a bit-shifting operation, which inherently reduces the hardware cost. The simulation results of the method with input data retrieved from the

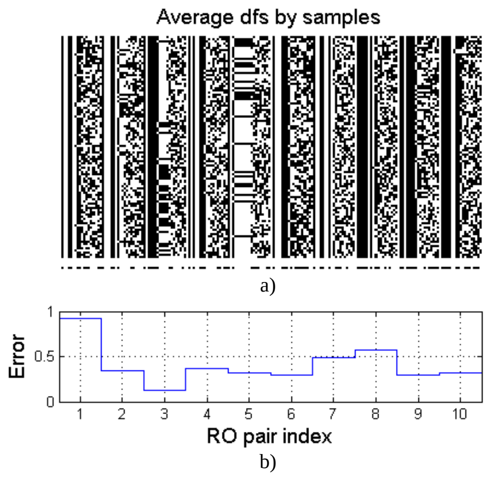


Figure 2. The bitmap image illustrating differential frequency samples (a) and errors to respected mean frequencies.

Table 1. Algorithm to calculate the averaging values of RO differential frequencies

Step	Task
1	Evaluate the RO array
2	Accumulate the df data
3	Assign $df_{j0} = df_{1j}$
4	Calculate $\Delta_i = df_{ij} - df_{j0}$, $i = \overline{1, n_{sample}}$
5	Calculate $\Delta = \sum_{i=1}^{n_{sample}} \Delta_i = \sum_{i=1}^{n_{sample}} df_{ij} - n_{sample} df_{j0}$
6	Calculate $df_{mean,j} = \frac{1}{n_{sample}} \sum_{i=1}^{n_{sample}} df_{ij} = \frac{\Delta}{n_{sample}} + df_{j0}$
7	Eliminate the variation part in step 6 to retrieve the averaging value $df_{mean,j}$.

experiment are shown in Figure 2, where the last line in the bitmap figure is the average df values, and the below plot is the errors between mean dfs which are arithmetic average values, and corresponding values extracted by the algorithm. These absolute error values less than 1, so confirm the accuracy of the method. Obviously, the averaging method needs an extra data processing unit to evaluate the algorithm in Table 1, which consumes more hardware resource. In addition, this technique also requires an interval of time to accumulate and process the data, which let the device be affected by external factors.

2.2. Extracting Stable Bit-string by Excluding Variation Bits in df Sample Values

The simplest way to extract the stable bits is excluding the variation portion of a RO differential frequencies. All the dfs data will be cut off toward the least significant bits by the length of the longest variation portion, denoted as N_{EX} , i.e. correspond to the worst case.

The bit-string is formed by concatenation the invariant parts of RO dfs (Figure 3). In fact, the length of fluctuated data in each df component are not the same. This irregularity is fundamentally the nature of process variations. The maximum and minimum deviations for the IC are 1.37×10^3 and 0.15×10^3 , respectively, correspond to the unstable portion from different df values alters from 9 to 13 bits.

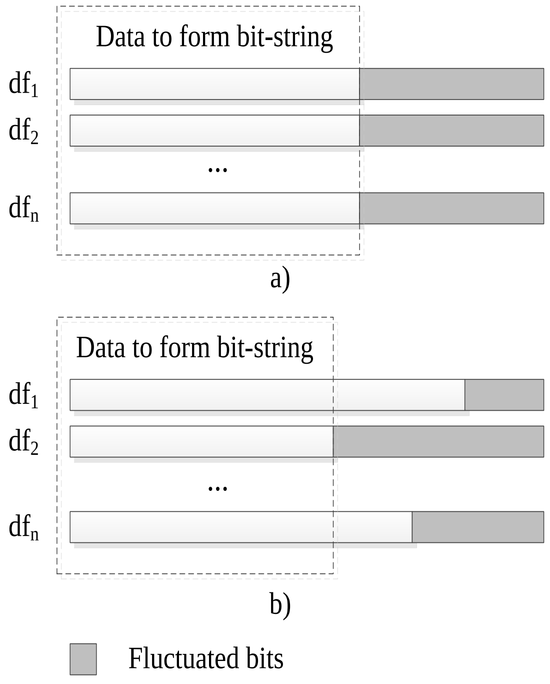


Figure 3. Method for obtaining the stable bit-string from invariant parts of dfs .

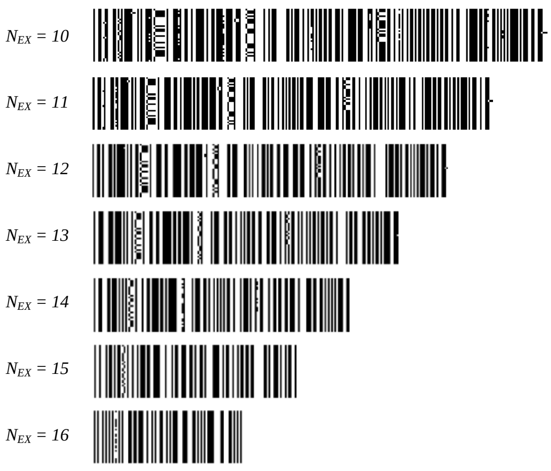


Figure 4. The bitmap image depicts the relationship between bit-string stability and the number of excluded bits.

N_{EX} could be theoretically approximated using the statistical parameters of df data [18]:

$$N_{EX} = \left[\log_2\left(6\sigma_{df_{mean},\max}\right)\right] = 13\left[bit\right]$$

We evaluate the stability of the bit-string using MATLAB simulation with different N_{EX} with results presented in Figure 4. This figure shows that, the stability of the bit-string increases when the number of excluded bits is increased, i.e the remaining df bit-string is decreased, so that the total length of the output bit-string is reduced. This method, evidently, is easy and low hardware resource consumption, but also has disadvantages. Firstly, it is inefficient in terms of information exploitation caused of the irregularity of df data fluctuation in practice. For df values that highly stable, i.e have a small fluctuation in bit-string, the amount of cut bits may be greater than the length of invariant portions. In contrast, for less stable values, a supposed excluding data may not be sufficient, particularly in the case of occurred outliers within the data that increase the bit-string disorder. Secondly, limiting values to a set of quantized values alters device ID vectors, which in turn affects PUF inter- and intra-distances. This issue results in the reliability of device authentication. Table 2 depicts this problem by simulation for 5 ICs with varying numbers of excluding bits. As a result, when increasing N_{EX} to the critical value of $N_{EX} = 17$, the minimum Euclidean distance between IC's nominal ID is reduced to 0, implying that extracted bit-strings cannot be discriminated. So the values of N_{EX} have the upper bound $N_{EX} = 16$. For N_{EX} in the range of $13 \le N_{EX} \le 16$ the minimum inter distances are still exceeding the threshold of $d_{thr} = 1.4 \times 10^{-3}$, which is equal to six times of the greatest value

Table 2. Distances between nominal IDs $\left[\times 10^{3}\right]$ of devices for different values of N_{EX}

	$N_{EX} = 13$				$N_{EX} = 14$				$N_{EX} = 15$			
	IC2	**IC3**	**IC4**	**IC5**	**IC2**	**IC3**	**IC4**	**IC5**	**IC2**	**IC3**	**IC4**	**IC5**
IC1	16.8	21.7	23.3	22.8	18.6	22.5	23.8	24.5	27.5	33.7	27.5	27.5
IC2		23.1	25.9	21.7		25.7	28.1	21.0		37.2	31.8	27.5
IC3			28.5	19.4			28.6	23.1			33.7	29.7
IC4				23.3				25.7				27.5
	$N_{EX} = 16$				$N_{EX} = 17$				$N_{EX} = 18$			
	IC2	**IC3**	**IC4**	**IC5**	**IC2**	**IC3**	**IC4**	**IC5**	**IC2**	**IC3**	**IC4**	**IC5**
IC1	31.8	22.5	38.9	22.5	0	0	63.5	0	89.8	0	0	0
IC2		38.9	38.9	22.5		44.9	77.8	44.9		89.8	89.8	89.8
IC3			44.9	31.8			63.5	0			0	0
IC4				31.8				63.5				0

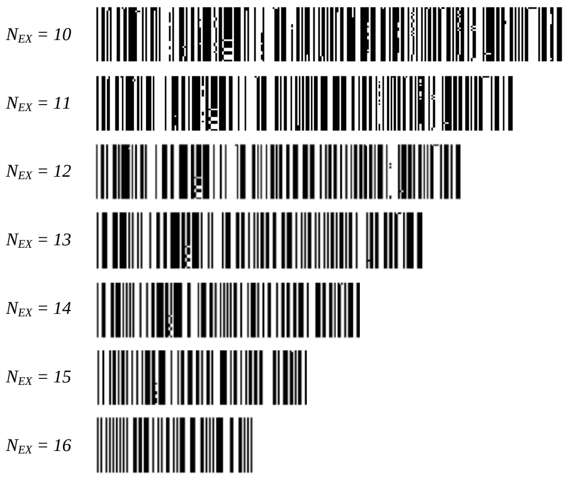


Figure 5. The bitmap image illustrating the simulation result for combination averaging df sample and bit-cutting method.

of the maximum intra distance's standard deviation plus the greatest value of the maximum intra distance's mean [17]. This is able to afford the success in device authentication. The length of the output bit-string is determined by the value of N_{EX}. The larger N_{EX} to be chosen, the sorter output bit-string to be obtained, and vice versa.

Due to outliers occurred during the operating process, the efficiency of them is scaled down. In the first step to improve the bit-cutting method, we add the data averaging stage before the main algorithms. The simulation of this method is presented in Figure 5, where the input of the process is mean df values instead of absolute df values. In comparison to the results in Figure 4, the output bit-string are more stable. The general bit-string is the combination of the partial bit-strings.

2.3. Extracting Stable Bit-string by Using Adaptive Data Mask

The major concept behind this approach is that since df sample values differ by the amount of least significant bits, we can obtain the invariant bit-string by applying the data mask to the df sample values.

Table 3 shows the algorithm used to create the data mask. The stable partial bit-string is updated from sample to sample by combining the value df_{j0}, $j=\overline{1,n_{ring}-1}$ and the mask through a procedure called the key extraction:

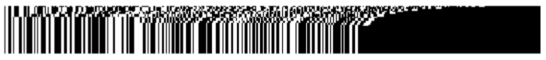
$$key0_{ij,k}=\begin{cases} df_{j0,k} & if \quad mask_{ij,k}=1 \\ 0 & if \quad mask_{ij,k}=0 \end{cases} \qquad (3)$$

Therefore, $mask_{ij,k}$ is k^{th} bit of the partial mask $mask_{ij}$ respected to i^{th} sample of RO_j differential frequency. The concatenation of the partial bit-strings results in the overall bit-string.

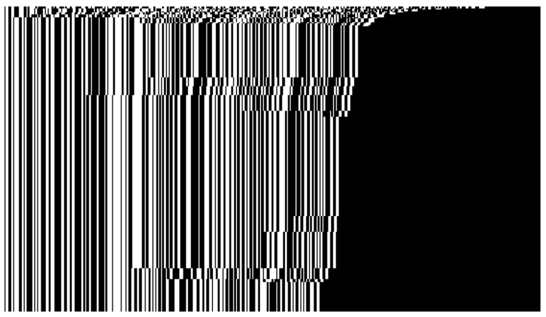
Table 3. Algorithm to create the data mask adapted to input differential frequency data

Step	Task
1	Evaluate the RO array
2	Accumulate the df data, which in form $(df)_{n_{sample}\times(n_{ring}-1)}$
3	For each RO_j, assign the first df sample df_{1j} to the reference value, in bit-string form: $df_{j0}=df_{1j}$
4	For each df_{ij} sample, calculate: $df_{xor\_ij}=df_{ij}\oplus df_{j0}$
5	Calculate the temporary masks: $mask\_temp_j=\bigcup_{i=1}^{n} df_{xor\_ij},\ j=\overline{1,n_{ring}-1}$ Filter out all the '0' bits in the maximum range limited by '1' bits[1] of $mask\_temp_j$ to receive $mask\_temp1_j$.
6	Invert the $mask\_temp1_j$ to receive partial mask respected to RO_j.

[1] This means that all intermediate values within the variation data range will be removed.



a) $n_{sample} = 32$



b) $n_{sample} = 200$

Figure 6. The bitmap image illustrating the convergence of ICs' raw bit-string to stable bit-string using dynamic masking method.

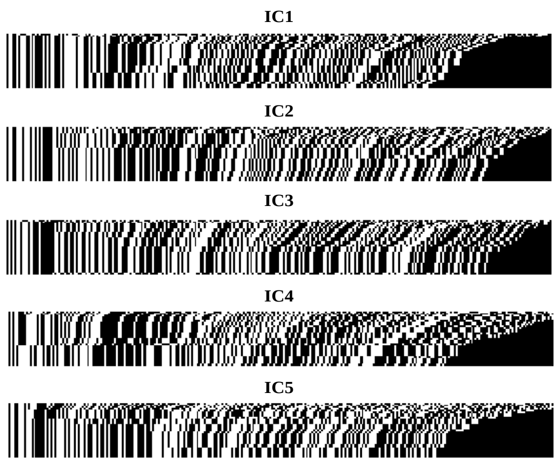


Figure 7. The bitmap image illustrating the simulation result of data masking process when integrating the averaging technique.

The main simulation results are presented in Figure 6 where each cycle of differential frequency averaging is assigned to a single row. The bit-string constructed by appending all of dfs contemporary stable parts converges to a single bit-string known as raw bit-string raw bit-string (the last row in the bitmap image). Because of raw bit-string contains a rightmost zero bit-string formed by the masking process, a number of intended bits are appended to the raw bit-string data to separate the required stable bit-string. We can take some brief reviews from the bit-string extraction progress shown in Figure 6 as below. Firstly, the algorithm used to form the mask adapts to the individual variations of dfs, allowing it to exploit more information than the previous method, which assumed that dfs had the same number of variation bits. Secondly, the scheme is extremely sensitive to outlier data. Any change in df data, even if it is only one bit, can affect the order of the last bit-string. This disadvantage will be overcome by combination this algorithm with averaging method and bit cut-off method. We integrate the averaging technique to the algorithm in Table 3 to condition the input data as presented in Table 4. The data masking process now is more stable as simulation results in Figure 7.

Table 4. Modification of the data masking algorithm

Step	**Task**
1	Evaluate the RO array
2	Accumulate the df data, which in form $(df)_{n_{key\_sample}\times(n_{ring}-1)\times n_{sample}}$
3	Calculate df_{mean} for all the RO pairs of all key samples
4	For each RO_j, assume the first $df_{1,mean,j}$ sample (respective to the first key_sample variable) is the reference value, in bit-string form: $df_{mean\_0,j} = df_{1,mean,j}$
5	For each $df_{k,mean,j}$ sample, $k = \overline{1,n_{key\_sample}}$, calculate: $df_{k,mean_{xor},j} = df_{k,mean,j} \oplus df_{mean\_0,j}$
6	Calculate the temporary masks: $mask\_temp_j = \bigcup_{i=1}^{n} df_{k,mean_{xor},j}$ Filter out all the '0' bit in the maximum range limited by '1' bits of $mask\_temp_j$ to receive $mask\_temp1_j$.
7	Invert the $mask\_temp1_j$ to receive partial mask respected to RO_j.

To further improve the efficiency of the adaptive data masking algorithm, we integrate the averaging technique and bit-cutting algorithm into it. The algorithm of this combination method is presented as follows. After step 3 of the algorithm as presented in Table 4, we add the step 3a: Cut rightmost bits in df_{mean} bit-string data by a number of excluded bits N_{EX}. The remaining steps are identical to the steps as in Table 4. The simulation of this combination variant is presented in Figure 8. As we can see, the convergence process to the unique bit-string is fast and more stable. This also makes the design more complicated and require higher resource usage.

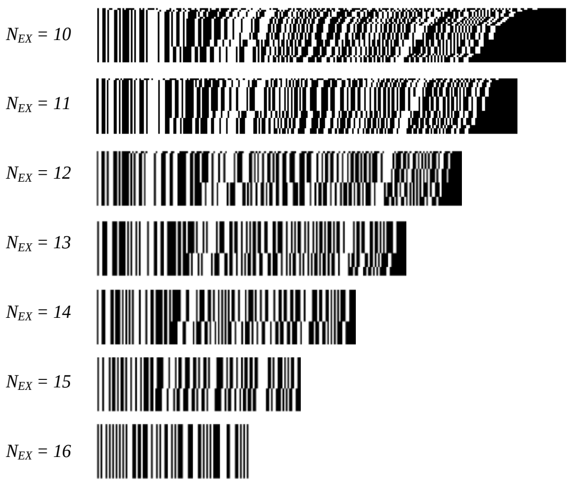


Figure 8. The bitmap image illustrating the simulation result for combination averaging df sample, bit-cutting and data-masking method.

2.4. Extracting Most Repeated Element from Statistical Distribution

This method is to prevent an emergence phenomenon in df value, which occur when the numbers of excluded bits are not large enough. The df values are cardinal numbers in natural, so any emergence df value will raise the bit-string changes, especially at boundary of power-by-2 values, e.g. $\{0111\}$ and $\{1000\}$. This method is not suitable for applying to absolute df sample values because they are finite values that different normally. Instead of this, we apply this method to average values of dfs after cutting off the variation bits. The number of excluded bits is evaluated from experiment to gain more information, especially in the cases of small fluctuated dfs. These values fall into a limited set of contemporary stable value samples. The most repeated sample is determined by analyzing the frequency distribution of the elements. The algorithm presented in Table 5, where n_{ic}, n_{key_sample}, and n_{sample} are the number of ICs, the number of key samples, and the number of samples, respectively. The remaining averaging df samples are combined to form the unique bit-string.

Table 5. Algorithm to extract the unique bit-string from the most repeated elements

Step	Task
1	Evaluate the RO array

2	**for** i = 1 **to** n_{ic}
	 for j = 1 **to** n_{key_sample}
	  for k = 1 **to** n_{sample}
	   Calculate the df_{mean} from df sample values.
	  end;
	  Extract $df_{1,mean}$ bit-string from df_{mean} by cutting off bits, write $df_{1,mean}$ to RAM.
	 end;
	end;

3	Calculate the frequency distributions of $df_{1,mean}$.
4	Extract the $df_{1,mean}$ samples that occur most, note as $df_{2,mean}$.
5	Concatenate $df_{2,mean}$ elements to form the key.

In order to investigate the efficiency of the algorithm, we simulate it with experimental data on Matlab software. The simulation results are presented in Figure 9, in which the output raw key is the last row of the image. This obviously indicates the convergence of the data to the value that occurred most frequently. Hence, we can take some notations about the method. Firstly, the number of data bins in the constructed histogram increase when we reduce the assumed number of unstable bits, and vice versa. This leads to large required df samples to exactly determine the most frequent value. An encryption or decryption protocol using the key extracted directly from this may repeat many times to prevent the wrong value in frequency distribution when the number of samples is not large enough.

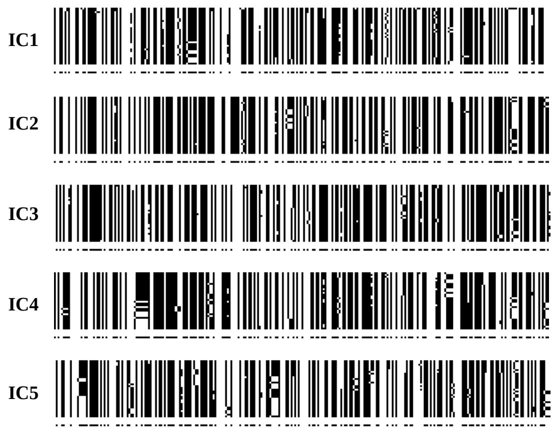


Figure 9. The bitmap image illustrating the key extraction by concatenation the most frequent elements of average *df* values.

The error occurred when there is a fluctuation in the practical data and the number of samples is not large enough, not by the algorithm itself. Secondly, when limiting the number of elements to achieve the more exact frequency distribution, the information may be lost so that the efficiency of the method is reduced. So the assumed number of unstable bits is trade-off between: i) The security of the extracted key, the required key length, device-primitive features; and ii) The evaluation time, hardware resource cost, and the implementation complexity.

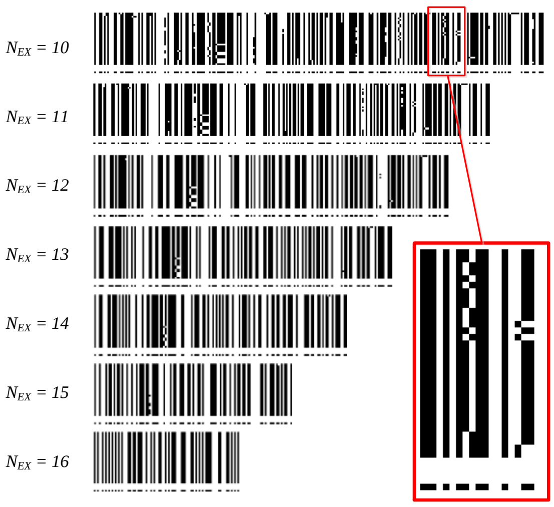


Figure 10. The bitmap image illustrating the simulation result for combination statistical distribution and bit-cutting methods.

When integrating the statistical distribution algorithm with bit-cutting method, the accuracy of the bit-string extraction is improved caused by the fact that the bins contain only values among the limited set of quantized values. The bit-cutting method similar to the quantization process, where df sample values are distributed to quantized levels. As we can see from simulation results in Figure 10, the disturbances occurred are eliminated to extract the stable bit-string (the last line in bitmap image).

3. Implementation of Stabilized Techniques on FPGA Device

The methods and techniques presented in the previous section work with RO PUF data that retrieved from experiments. However, for the purpose of on-chip examination, we need to implement the designs that perform stabilized methods and algorithms. We have preliminary implemented following techniques on FPGA Artix-7.

- Bit-cutting combined with sample averaging method,
- Data masking method.

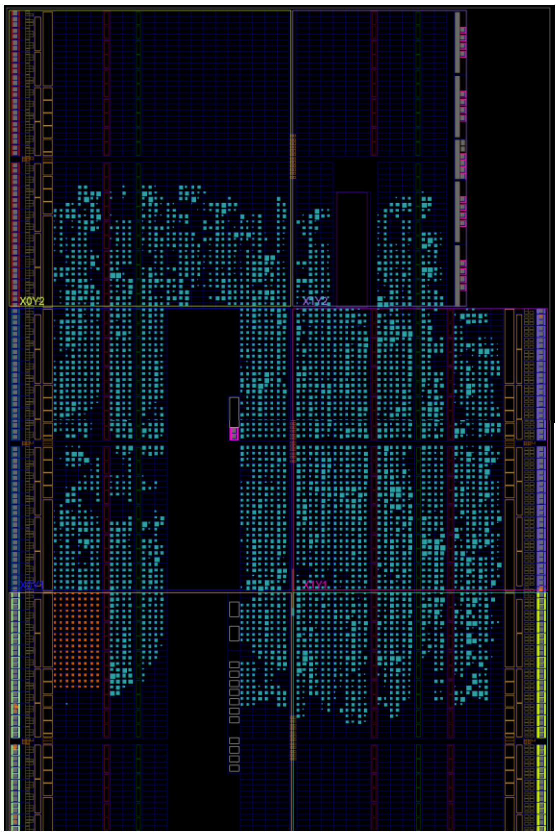


Figure 11. Layout of output bit-string stabilized design by data masking method on FPGA Artix-7.

Table 6. Hardware resource consumption of stabilized design by bit-cutting combined with sample averaging method, implemented on FPGA Xilinx Artix-7 XC7A35T

Resource	Utilization	Available	Utilization (%)
LUT	980	20,800	4.71
FF	630	41,600	1.51
BRAM	1	50	2.00
IO	9	170	5.29

RO arrays are similar to differential frequencies circuit from ID extraction scheme in [17]. Layout of data masking design is presented on Figure 11, and hardware resource consumption of the designs are presented on Table 6 and Table 7, respectively.

Table 7. Hardware resource consumption of stabilized design by data masking method, implemented on FPGA Xilinx Artix-7 XC7A35T

Resource	Utilization	Available	Utilization (%)
LUT	10,608	20,800	51.00
FF	10,480	41,600	25.19
BRAM	256	50	2.67
IO	8	170	4.71

Output bit-strings retrieved from multiple times of hardware evaluation are very stable while maintain high uniqueness, which can be used for further security applications.

4. Conclusion

In terms of implementation, FPGA-based RO PUFs have gained popularity due to benefits such as reasonable complexity, resource efficiency, and so on. RO PUFs consist of self-oscillating schematics whose output frequency is primarily determined by the non-identical delays of logical components due to the inherent mismatch of physical devices from manufacturing process. These fluctuations are device-primitive and satisfy many PUF prerequisites, including unpredictability, uniqueness, and reliability. As a follow-up to our previous work on using FPGA-based RO PUFs to extract a device's ID, we focused on using techniques to extract unique and stable bit-strings that could be used in secure key generation for cryptographic systems.

In this work, we have proposed some algorithms and techniques for dynamically extracting unique and stable bit-strings from differential frequencies data of RO PUFs. The algorithms are validated using MATLAB simulation and implemented on Artix-7 FPGA devices. The advantages as well as the drawbacks of each scheme have been examined. In addition, the implemented designs that combine methods work consistently and efficiently. Since the extracted bit-strings are unique, unpredictable, and do not require a mechanism for storage and distribution, they can be used in secure cryptographic systems as well as other specific hardware security applications.

Acknowledgements

This research is funded by Vietnam National Foundation for Science and Technology Development (NAFOSTED) under grant number 102.02-2020.14.

This work was partly presented at the ICTC 2021 conference [18].

References

[1] G. E. Suh, S. Devadas, Physical Unclonable Functions for Device Authentication and Secret Key Generation, 44th ACM/IEEE Design Automation Conference, 2007, pp. 9-14.

[2] R. Maes, Physically Unclonable Functions: Constructions, Properties and Applications, Springer Science & Business Media, 2013.

[3] V. Vivekraja, L. Nazhandali, Circuit-Level Techniques for Reliable Physically Uncloneable Functions, 2009 IEEE International Workshop on Hardware-Oriented Security and Trust, 2009, pp. 30-35.

[4] A. Maiti, P. Schaumont, Improved Ring Oscillator PUF: An FPGA-Friendly Secure Primitive, Journal of cryptology, Vol. 24, No. 2, 2011, pp. 375-397.

[5] M. Gao, K. Lai, G. Qu, A Highly Flexible Ring Oscillator PUF, Proceedings of the 51st Annual Design Automation Conference, 2014, pp. 1-6.

[6] F. Kodýtek, R. Lórencz, J. Buček, Improved Ring Oscillator PUF on FPGA and Its Properties, Microprocessors and Microsystems, Vol. 47, 2016, pp. 55-63.

[7] C. Bösch, J. Guajardo, A. R. Sadeghi, J. Shokrollahi, P. Tuyls, Efficient Helper Data Key Extractor on FPGAs, International Workshop on Cryptographic Hardware and Embedded Systems, 2008, pp. 181-197.

[8] Z. Paral, S. Devadas, Reliable and Efficient PUF-Based Key Generation Using Pattern Matching, 2011 IEEE International Symposium on Hardware-Oriented Security and Trust, 2011, pp. 128-133.

[9] R. Maes, A. V. Herrewege, I. Verbauwhede, PUFKY: A Fully Functional PUF-Based Cryptographic Key Generator, International Workshop on Cryptographic Hardware and Embedded Systems, 2012, pp. 302-319.

[10] V. T. Tran, Q. K. Trinh, V. P. Hoang, Enhanced ID Authentication Scheme Using FPGA-Based Ring Oscillator PUF, IEEE 13th International Symposium on Embedded Multicore/Many-core Systems-on-Chip (MCSoC), 2019, pp. 320-327.

[11] N. H. Weste, D. Harris, CMOS VLSI Design: A Circuits and Systems Perspective, Pearson Education, India, 2015.

[12] O. Mencer, D. Allison, E. Blatt, M. Cummings, M. J. Flynn, J. Harris, S. Shirazi, The History, Status, and Future of FPGAs: Hitting A Nerve with Field-Programmable Gate Arrays, Queue, Vol. 18, No. 3, 2020, pp. 71-82.

[13] W. Stallings, Cryptography and Network Security Principles and Practices, 4th Edition, Pearson Education, London, 2006.

[14] J. Delvaux, D. Gu, D. Schellekens, I. Verbauwhede, Helper Data Algorithms for PUF-Based Key Generation: Overview and Analysis, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol. 34, No. 6, 2015, pp. 889-902.

[15] J. Delvaux, Security Analysis of PUF-Based Key Generation and Entity Authentication, Shanghai Jiao Tong University, China, 2017.

[16] H. Kang, Y. Hori, T. Katashita, M. Hagiwara, K. Iwamura, Cryptographic Key Generation from PUF Data Using Efficient Fuzzy Extractors, 16th International Conference on Advanced Communication Technology, 2014, pp. 23-26.

[17] V. T. Tran, Q. K. Trinh, V. P. Hoang, A Robust Euclidean Metric Based ID Extraction Method Using RO-PUFs in FPGA. Integration, Vol. 82, 2022, pp. 37-47.

[18] V. T. Tran, Q. K. Trinh, V. P. Hoang, Stabilizing On-chip Secure Key Generation Using RO-PUF, 2021 International Conference on Information and Communication Technology Convergence (ICTC), 2021, pp. 805-809.