Highly Secure Data Encryption Devices Using Unique Physically Unclonable Key

Van-Toan Tran, Quang-Kien Trinh*, Tri-Hieu Le, Tung-Lam Nguyen, Van-Phuc Hoang
Le Quy Don Technical University, Hanoi, Vietnam
*Correspondence: kien.trinh@lqdtu.edu.vn

Abstract **- Physically Unclonable Functions (PUFs) exploit intrinsic mismatch of physical devices to form device-specific data that uniqueness, reliability and unpredictable. PUF find important applications in hardware security as identification and authentication, secure key generation. In this work we proposed a crypto key extraction scheme using RO-PUF, that can be used for critical security applications. Specifically, we practically demonstrated that the extracted key can be used for data encryption devices by AES cipher. The encryption devices hence become unique and physically unclonable. The encrypted data is highly secure in the sense that the key is intrinsically stored inside the device is theoretically unknown even to direct owners.**

Keywords - RO-PUF, FPGA, hardware security, key generation.

I. INTRODUCTION

Physically Unclonable Functions (PUFs) is a technique that has been developed for two recent decades. At a glance, PUF would be described as "an object's fingerprint" [1]. Due to many advantages, PUF has been employed in hardware security applications such as device identification and authentication, secure key generation, IP protection, etc. In this work, we concentrate on techniques and algorithms to apply PUF in secure key generation.

In practice, the PUF response samples are not identical in distribution and could not be re-constructable in evaluation. So, they are suited for device authentication, but not possible to be used directly as a secure key. Authors in [2] have proposed an ID extraction and authentication scheme based on RO-PUF in which differential frequencies instead of the absolute frequencies could effectively exclude the global variation factors. However, response samples are still unstable due to intrinsic fluctuations as presented in Fig. 1.

The common requirements for a cryptography application are securely generating, accumulating, and distributing the keys, which are far from trivial satisfied by hardware solutions [1]. To deal with the problems, many PUF-based key generation models are proposed. In [4], the authors proposed a scheme using error correction encoding (ECC) to maintain the stable PUF output even in unstable operating conditions. The authors in [5] designed a novel variant of the RO-PUF based on Lehmer-Gray order encoding and a resource-optimized BCH decoder. The latter is the main component of the helper data scheme, which produces the keys through a procedure called entropy accumulator. In [6], the authors replaced replacing the hash function by the BCH to improve the efficiency of the fuzzy extractor. Nonetheless, most of the above approaches have been proposed for the conventional RO-PUF that retrieved the bit-string from the sign function [7], which is quite an inefficient design. In addition, those key generation schemes employed quite complex and consumed intensive resources from the hardware implementation point of view. Furthermore, none of these works practically shows and verifies the application of PUF key in reality.

In this work, we propose a stable secure key generation design based on RO-PUF implemented on FPGA devices. Using a very lightweight circuit, the proposed design could extract the unique and stable key for direct use in the AES encryption engine. The design has been successfully verified on Xilinx FPGA devices and is ready for further deployment in more sophisticated security applications.

The rest of the paper is organized as follows. Section II provides an overview of the RO-PUF design and its implementation on FPGA. Next, in Section III, we present the method to extract the stable key from RO-PUF data, as well as evaluate the stability of the extracted bit-string. Section IV presents an application model of generated keys in AES cryptography. Finally, Section V concludes the paper.

II. IMPLEMENTATION OF RO-PUF DESIGN ON FPGA DEVICES

Our RO-PUF is based on the conventional RO-PUF scheme of Suh and Devadas [3] and ID extraction and authentication scheme [2] with some modifications for implementation on Xilinx Artix7 FPGAs. The functional schematic of RO-PUF is shown in Fig. 2. Basic RO comprises *N* inverters and a NAND gate. In our design, an inverter occupies one LUT primitive in Xilinx Artix-7 FPGA. These elements are floor-planned and routed manually to maintain a regular and symmetric layout. RO frequencies are measured by counters that are evaluated simultaneously to reduce the data processing time. The whole design is kept compact and explicit in order to maintain the stability and reliability.

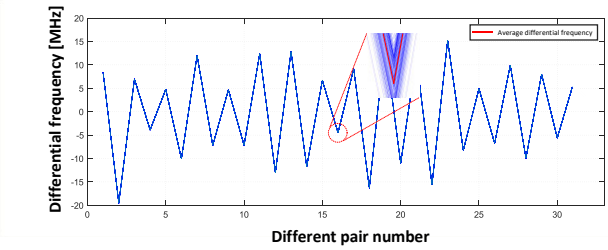


Fig. 1: ID vector comprised of differential frequencies from array of 32 ROs, evaluated by 256 samples (df data in 24 bit).

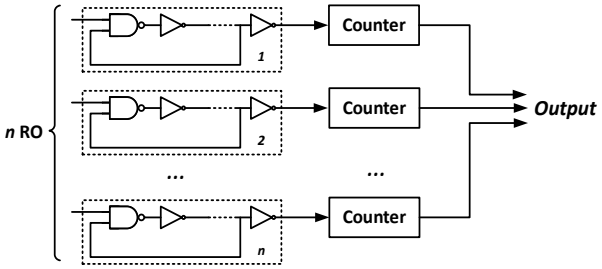


Fig. 2: Proposed RO-PUF functional schematic.

Apart from intrinsic device variations, the other important factors affecting RO-PUF frequencies include the fluctuation of ambient temperature and supply voltages. This leads to the instability in RO-PUF responses, which can be modeled as [1]

$$f_{RO} = f_{nominal} + \Delta f_{proc,local} + \Delta f_{proc,global} + \Delta f_{OP} \quad (2)$$

Therein, $f_{nominal}$ is the nominal RO frequency measured for the same type of devices under the nominal condition, in this case, is ambient temperature 25^0C; core voltage $1.0\,V$. The random variables $\Delta f_{proc,global}$ and $\Delta f_{proc,local}$ are the frequency deviations due to global variation and local variation factors, respectively. Δf_{OP} is the frequency deviation caused by operation conditions.

In this particular RO-PUF design, we implemented an array of 32 ROs, each RO consists of 16 inverters and a NAND gate. Data retrieved from 4096 evaluation is transmitted to PC for processing. The RO frequency value is measured repeatedly by multiple times for 32×5 ROs under fixed operating conditions (25^0C; 1.0 V). The RO frequency varies due to the stochastic fluctuation in ambient temperature and the supply voltage. For a single RO, this fluctuation is corresponding to the Δf_{OP} variation. This impact is quantified by the temporal variation coefficient σ/μ or reliability factor [2] ($1 - \sigma/\mu$) (in percentage), in which μ (σ) is the mean value (standard deviation) of sampled frequency. Fig.3 shows a histogram of frequency samples respected to RO5/IC1from 4096 evaluations. The mean frequency is 102.460 MHz and the standard deviation is 60.235 kHz, corresponding to temporal variation coefficient ~ 0.06% (the expected value is less than 1% according to [1]).

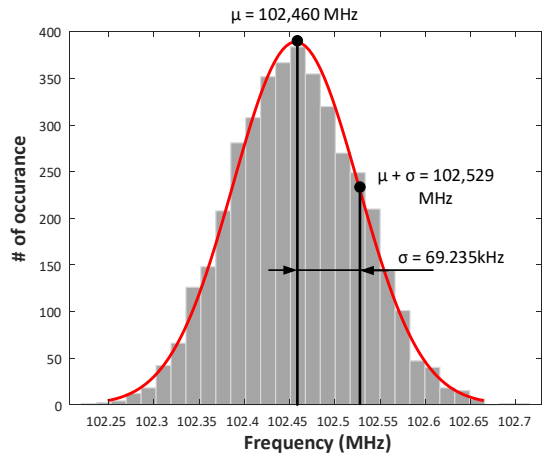


Fig. 3: Histogram of RO frequencies (RO5/IC1) retrieved from 4,096 repetitive measurements.

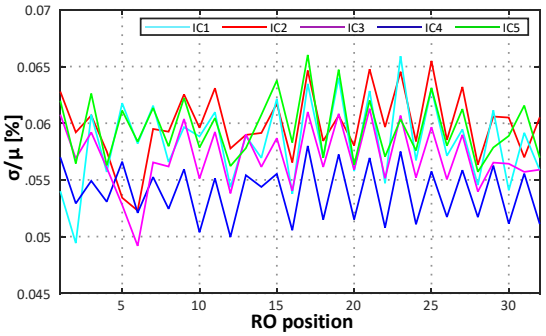


Fig. 4: σ/μ ratio of 32 ROs from 5 ICs, evaluated from 4096 samples at the ambient temperature of 25^0C.

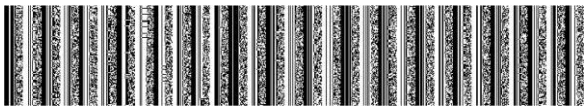


Fig. 5: Bitmap image of 256 differential frequency samples.

The measured temporal variation coefficients of 5 × 32 RO are presented in Fig.4, which are retrieved from 32 RO of 5 FPGAs. From the estimation, the temporal variation coefficients have the minimum value 0.0491% (RO6/IC3) and maximum value 0.0661% (RO17/IC5), corresponds to the reliability 99.95% và 99.93%, respectively. This result indicates that the temporal variation has little impact on the RO frequencies and can be ignored in further analysis.

To exclude the effect of global variations, we divide ROs by pairs. Each RO pair consists of two successive ROs. We use differential frequencies from RO pair output as the RO-PUF responses [2] to remove the effect of global variations. Due to local variations and other factors, there are small fluctuations in differential frequency samples, i.e., samples different from each other from evaluations. This is shown in Fig. 5 for the bitmap representation of the extracted samples. It can be seen that the low significant bits flip more regularly. So, the direct usage of these data samples as a key is inappropriate. The directly extracted bit-strings are unstable and impossible to be used as a part of secure keys. In the following section, we will present the method that performs in-device stabilizing and extracting the unique stable bitstring. We also investigate the quality and different aspects of the extracted bit-strings to be used for the key generation process.

III. EXTRACTING UNIQUE AND STABLE BIT-STRINGS FROM UNSTABLE DIFFERENTIAL PUF FREQUENCY SAMPLES

In this section, we present a method to extract the stable key from the differential frequency samples. In this particular design, 32 ROs corresponds to 31 differential values that form a vector representing the intrinsic local variation. To ensure the accuracy of the measured samples, we set a counting period of 20 ms for accumulating the samples. The maximum error in frequencies is 50Hz, which corresponds to relative measurement error and accuracy are 0.002 % and 99.998 %, respectively. Details can be found in [8].

The overall key extraction procedure is shown in Fig. 6. The RO differential frequencies are calculated on-chip by a subtractor array located at the output of the RO array. A stabilizer is used to stabilize the differential frequency data to extract stable and unique bit-string, that is used to generate a secure key by a Hash function. This design has been described in VHDL and synthesized with Xilinx Artix7 XC7A35T FPGA devices using Xilinx Vivado 2019.2 tool.

In this work, we propose to stabilize the sample vector by the averaging method. As we will show later, the particular differential sample values are highly sensitive to the design itself. For example, moving the RO location will or any other component in auxiliary circuitry could affect the extracted values. Nonetheless, once the design is fixed, the temporal fluctuations are typically small as we have investigated earlier (i.e., frequency temporal fluctuation of ~0.06%).

In a deeper analysis, from the measurement values, we found that the response bit-string samples are not identical, as well as the number of fluctuated bits, are not the same, as presented in Fig. 7a. By the straightforward approach we extract the stable bits for the worst case (i.e., respected to the longest number of fluctuated bits) as shown in Fig. 7b. From examining the response samples, it can be seen that the number of unstable bits falls in the range of 9 to 13 bits. The stability of the bit-strings increases when we enlarge the number of excluded bits (N_{EX}). For the worst case, when we cut off the bit-string samples with $N_{EX} = 15$, the output bit-strings are completely stable, as presented in Fig. 8. In such case, from 32 ROs, we could successfully extract stable vectors of 136 bits. This observation indeed is an inevitable tradeoff between the length of the extracted bit-strings and their stabilities. Compromising the bit length could enhance the stability but that may degrade the uniqueness. This is because among the physical devices, the local frequencies are highly correlated as reported in [2]. Nonetheless, the extracted bit-string is not directly used for the key but as input for the Hash function. Therefore, as long as the extracted bit-strings are stable and unique, they are suitable for secret key generation.

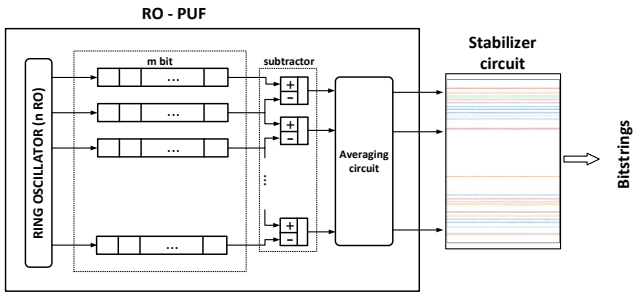


Fig 6: Bitstrings generation and stablizing process process.

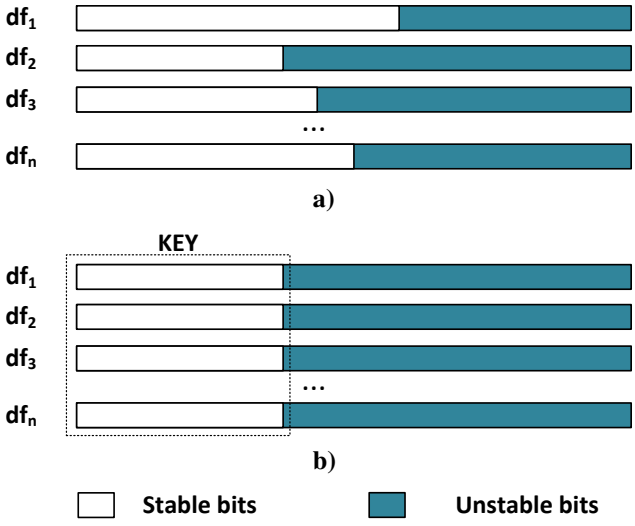


Fig 7: (a) An example distribution of stable and unstable bit in vector of differential frequency, (b) extracting stable bits by cutting the longest unstable bits.

In practice, to ensure high stability and a long enough number of bits, we proposed to stabilize the bitstring by taking the average of a large number of samples. To avoid expensive division operation, we take 2^M samples so that the averaging is simplified to be cutting M bits from the accumulated value. We have synthesized the design and implemented on the Xilinx Artix-7 (XC7A35T) FPGA devices, where for each device, bitstrings are extracted for 4 different RO locations as shown in Fig. 9. The hardware utilization is significantly improved in comparison to proposed designs (TABLE 1). Specifically, our proposed design does not require the block RAM (BRAM) for helper data as in [6] that utilizes 38 block RAM modules for correction algorithm. For the examination of 4,096 (i.e., M = 10) samples, the number of stable output bit-string for sample averaging is 238 which is 1.7X longer, compared to the bitstrings extracted without averaging technique.

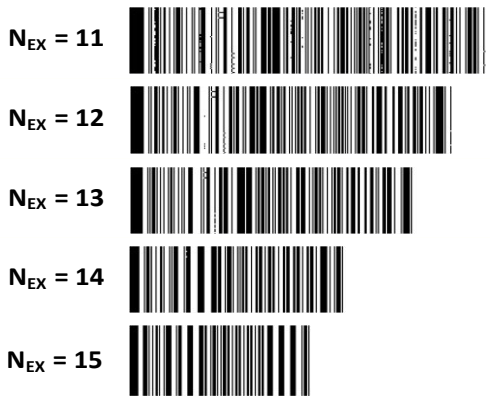


Fig 8: Dependence of the output bitstring stability on the number of excluded bits.

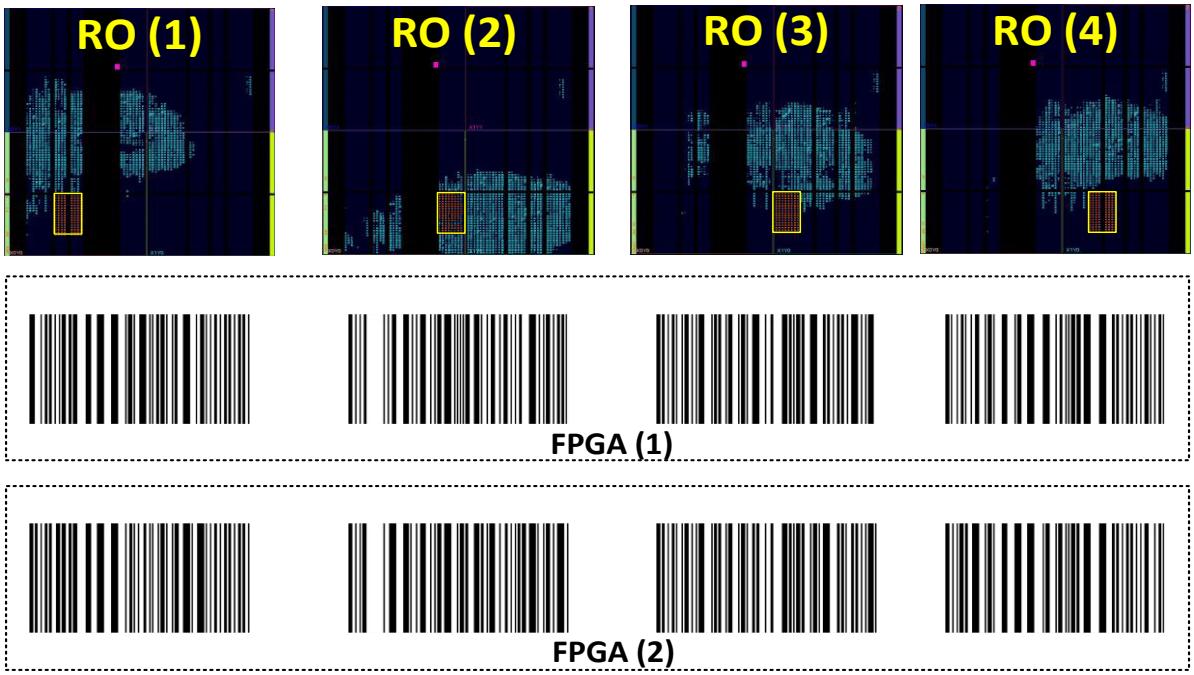


Fig. 9: Output bit-strings retrieved from the implementation of key generation on two FPGA devices (Xilinx XC7A35T FPGA).

TABLE 1: HARDWARE RESOURCE CONSUMPTION FOR IMPLEMENTATION OF THE KEY GENERATION DESIGN ON ARTIX-7 FPGA DEVICE

Resource type	Used	Available (XC7A35T)	Percentage used (%)
LUT	3798	20800	18.26
LUT-RAM	256	9600	2.67
FF	5301	41600	12.74
IO	10	170	5.88
BUFG	1	32	3.13

For the stability of the generated bit-strings, we examined for repetitive 1024 bit-strings from the devices for different locations and two FPGA devices (see Fig. 9). As the result reported in bitmap figures, the extracted bit-strings are completely stable, satisfy the requirements of a secure key, and can be used on-chip for a data encryption model.

IV. THE ON-CHIP SECURE KEY FOR DATA ENCRYPTION

In the proposed data encryption scheme as shown in Fig. 10, we directly use the considered stable bit-string as a seed value for the HASH function. The output from Hash function will be used as the section key for the AES cipher engine. The hardware card communicates with PC via an in-house software, that configs the mode of operation and performing data transfer for functional checking. This can be used as the end user software for the practical applications.

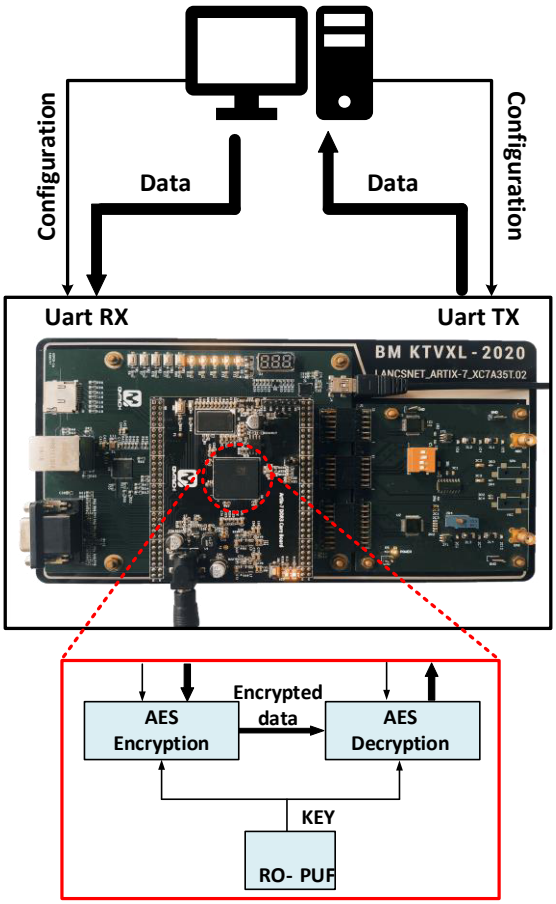


Fig. 10: AES cipher device implementation with built-in PUF key generation circuit.

For the purpose of demonstration, the input data that is converted from a binary image is encoded by software into a 128-bit bitstream without the bitmap headers to feed to AES encryption. The encrypted data is then converted back to BMP format for convenient viewing and checking. The whole process is shown in Fig. 11. In the following we conduct a verification of the proposed secure device for several testing scenarios:

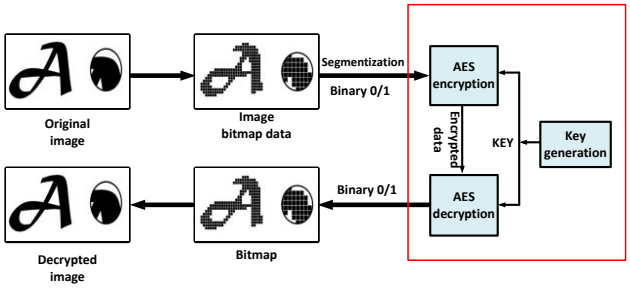


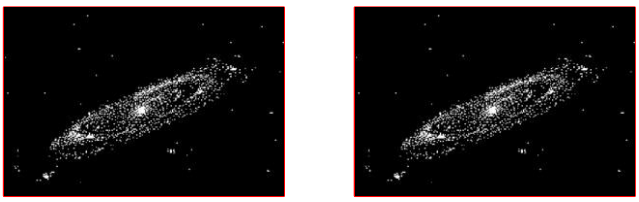
Fig 11: AES encryption and decryption.

Test scenario 1: Stability test

Using keys generated from the same position of RO array of one FPGA device and repeat the encryption/decryption for 16 times (Fig.12). The results show that input image and output image after decryption are identical for all trials. This demonstrate practically proven the stability of the generated keys.

Test scenario 2: Inter-die uniqueness test

In this test, we use the key generated from the first FPGA device to encrypt data. Then the encrypted data is decrypted by the key generated from the second FPGA device to decrypt (Fig. 13). As the results, the output decrypted data becomes almost a white noise image, that means the decryption key is not the same as the encryption key used. Hence, the results have confirmed the uniqueness of the extracted bit-strings when being examined the same design for different physical devices.

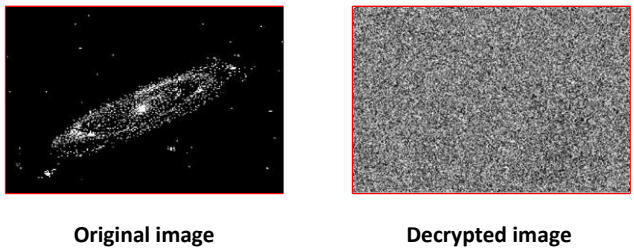


(a) (b)

Fig 12: Test scenario 1: (a) Original image (b) the decrypted image using the key extracted from same devices with the same RO position.

Test scenario 3: Intra-die uniqueness test

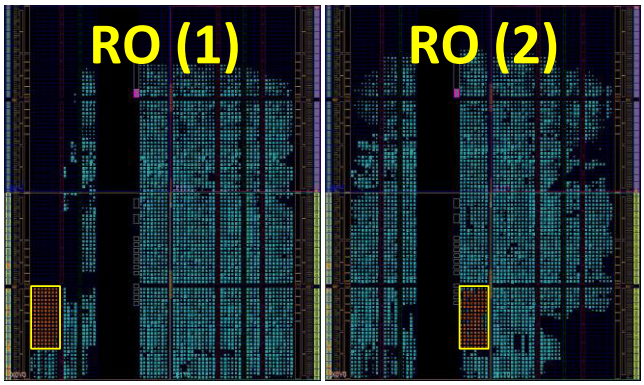
In this testing scenarios, we use only one FPGA devices but the ROs are intentionally placed in different position inside the chip as illustrated in Fig. 14. The data is encrypted by the RO at position 1 and then the retrieved data is decrypted by the two keys: one is generated at the RO position 2 and one by the same RO position 1. The results in Fig. 15 indicate that if the key extracted from the same RO position is used, the bitmap is recovered correctly (top image in Fig. 15).



(a) (b)

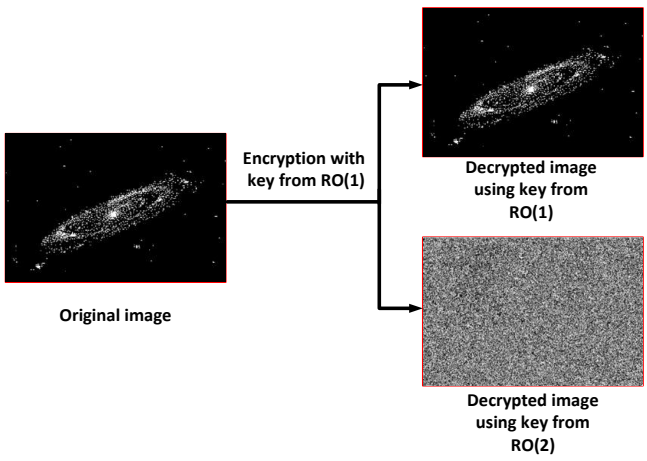
Fig 13: Test scenario 2: (a) Original image (b) the decrypted image using the key extracted from the other device with the same the same RO position.

On the other hand, when the key from other RO position 2 is used, the decrypted image becomes a white noise, i.e., like the result from the second test. Positions are distinguishable. The optimization of design process make change to the key generation, so it is essential to fix the positions of the RO array. The advantage of the on-chip key generation is that the keys are not disclosed, neither adversary nor re-manufactured.



(a) (b)

Fig 14: Positions of (a) RO (1) and (b) RO (2).



(a) (b)

Fig. 15: Test scenario 3: (a) Original image (b) the decrypted image using the key extracted from the same RO position 1 (top) and by the key extracted from the design with different RO position 2 (bottom).

V. CONCLUSIONS

FPGA-based RO-PUF is a powerful technique to implement the hardware security devices. In this work, we have proposed a lightweight solution that could exploit the advantages of RO-PUFs to generate stable and physically unclonable key generation. The design has been practically proven on FPGA devices. The keys extracted are satisfied the strict criteria in intra- and inter-die uniqueness and have shown a great stability. The keys are not only unpredictable but also unknown even to the device owner that permit us to further deploy more sophisticated and advanced hardware security application. In addition, the proposed designs provide the advantages of compactness and simplicity and are ready for being ported to other FPGA vendors or on ASIC.

ACKNOWLEDGMENT

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

REFERENCES

[1] Roel and M. A. E. S, "Physically unclonable functions: Constructions, properties and applications", Katholieke Universiteit Leuven, Belgium, 2012.

[2] R. Maes, *Physically Unclonable Functions*, Springer, 2013.

[3] Suh, G. E. and S. D. , "Physical unclonable functions for device authentication and secret key generation.", 44th ACM/IEEE Design Automation Conference IEEE, 2007.

[4] S Gubner and John A, *Probability and random processes for electrical and computer engineers*, Cambridge University Press, 2006, p. 138–183.

[5] Maes, Roel, A. V. Herrewege and I. Verbauwhede, "PUFKY: A fully functional PUF-based cryptographic key generator", International Workshop on Cryptographic Hardware and Embedded Systems. Springer, Berlin, Heidelberg, 2012.

[6] Delvaux, Jeroen and e. al, "Helper data algorithms for PUF-based key generation: Overview and analysis", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 34.6, 2014, pp. 889-902.

[7] Delvaux and Jeroen, "Security analysis of PUF-based key generation and entity authentication", Ph. D. dissertation, 2017.

[8] V. T. Tran, Q. K. Trinh and V. P. Hoang, "A robust Euclidean metric based ID extraction method using RO-PUFs in FPGA", Integration, Elsevier, vol.82, pp.37-47, Jan. 2022.