future internet

MDPI

Article

A Unified PUF and Crypto Core Exploiting the Metastability in Latches

Ronaldo Serrano *, Ckristian Duran, Marco Sarmiento, Tuan-Kiet Dang, Trong-Thuc Hoang and Cong-Kha Pham

Department of Computer and Network Engineering, The University of Electro-Communications (UEC), Tokyo 182-8585, Japan

* Correspondence: ronaldo@vlsilab.ee.uec.ac.jp

Abstract: Hardware acceleration of cryptography algorithms represents an emerging approach to obtain benefits in terms of speed and side-channel resistance compared to software implementations. In addition, a hardware implementation can provide the possibility of unifying the functionality with some secure primitive, for example, a true random number generator (TRNG) or a physical unclonable function (PUF). This paper presents a unified PUF-ChaCha20 in a field-programmable gate-array (FPGA) implementation. The problems and solutions of the PUF implementation are described, exploiting the metastability in latches. The Xilinx Artix-7 XC7A100TCSG324-1 FPGA implementation occupies 2416 look-up tables (LUTs) and 1026 flips-flops (FFs), reporting a 3.11% area overhead. The PUF exhibits values of 49.15%, 47.52%, and 99.25% for the average uniformity, uniqueness, and reliability, respectively. Finally, ChaCha20 reports a speed of 0.343 cycles per bit with the unified implementation.

Keywords: ChaCha20; PUF; RISC-V

Citation: Serrano, R.; Duran, C.; Sarmiento, M.; Dang, T.-K.; Hoang, T.-T.; Pham, C.-K. A Unified PUF and Crypto Core Exploiting the Metastability in Latches. *Future Internet* **2022**, *14*, 298. https://doi.org/10.3390/fi14100298

Academic Editors: Agostino Forestiero and Mohamed Abd Elaziz

Received: 30 August 2022
Accepted: 12 October 2022
Published: 17 October 2022

1. Introduction

A secure system's root of trust (RoT) requires some primitives to guarantee a minimum level of security [1–3]. For example, true random number generators (TRNG), physical unclonable functions (PUF), and cryptography algorithms can be used in a system's key generation and booting processes. Typically, each secure primitive is implemented in a standalone peripheral in the system, increasing the area overhead. In addition, a performance reduction is generated by moving the key into the respective usage module. Therefore, an approach consisting of unifying such modules with hardware security primitives is adopted to reduce the area overhead, exploiting different physical phenomena in the original peripheral. For example, a static random access memory (SRAM) can be used to generate a TRNG-PUF, exploiting the bit line's time response and leakage current [4–6]. In addition, a TRNG and a non-volatile random access memory (NVRAM) can be unified, exploiting the metastability in the sense amplifier [7] or reading the bit cell noise [8]. On the other hand, a PUF-NVRAM is unified using the initial charges into the floating gates' bit cells [9]. Another method consists of the unification of a TRNG-crypto core based on the metastability response using hold violations [10].

The static and dynamic entropies used in PUFs and TRNGs, respectively, are generated with different physical phenomena. Additionally, depending on the physical phenomena, the entropy source of the PUFs and TRNGs needs calibration or the utilization of particular techniques in the implementation to reduce the undesirable effects. For example, the metastability in flip-flops (FFs) can be used to generate a TRNG using two clocks with different frequencies [11–13]. In addition, a clock manager can generate a PUF, calibrating the clock frequencies to obtain a stable and unique response [14]. In another way, a TRNG based on frequency collapse can be implemented with a noise generator to guarantee that physical phenomena occur [15]. However, the mismatch causes an increase in the

static entropy, generating a stable and unique response in the physical phenomena of the PUF implementation [16]. Therefore, some strategies have been proposed to mitigate the systematic mismatch to generate a robust TRNG [17,18]. Furthermore, the implementation of an FPGA introduces an additional mismatch, increasing the implementation difficulty in the entropy source [19]. On the other hand, the cross-coupled inverter pair [20] and the NAND [21,22] latches are used in many TRNG implementations. However, TRNGs based on the metastability of latches require a calibration step due to the static entropy generated by the mismatch. Because of the many potential problems caused by the mismatch in the TRNG application, a calibration step or technique is mandatory in the implementation. Nevertheless, the mismatch can be exploited to generate a PUF implementation without a calibration step, reducing resource utilization.

Cryptography algorithms are used in data transmission, providing a secure end-to-end channel for vital information. However, side-channel attacks affect software implementations [23–25]. Therefore, new approaches are proposed to mitigate the vulnerabilities using in-memory computation [26] or hardware implementations [27–29]. The cipher most used in Transport Layer Security (TLS) is the advanced encryption standard (AES) [30]. Nonetheless, a new approach is proposed in TLS version 1.3, generating another alternative to the AES cipher [31,32]. The new cipher is the ChaCha20 based on Salsa20 [33], using 20 rounds in the operational matrix.

This work presents a unified PUF-ChaCha20 crypto core in an FPGA implementation. The main contribution of the current work is the unification of the PUF and crypto core without a significant area overhead, providing a new security approach in internet security in the one-time key (OTK) generation of the ChaCha20-Poly1305 AEAD into TLS version 1.3. Additionally, part of this work involves the determination of constraints and strategies for implementing a PUF without a calibration step. The static entropy in the implementation generates a PUF application using the unbalanced latches caused by the mismatch variations. In addition, the latches are used in one internal state of the ChaCha20 core without a calibration step, reducing resource utilization. The PUF presents, on average, 49.15% uniformity, 47.52% uniqueness, and 99.25% reliability in nominal conditions. The crypto core occupies 2416 LUTs and 1026 FFs in an FPGA implementation. Finally, the ChaCha20 with an internal PUF can generate a salted key using the static entropy created by the metastability in latches.

The remainder of this paper is organized as follows. Section 2 describes the utilization of the new approach of the unified PUF-crypto core in TLS. Section 3 discusses the architecture and the analytical model of the implemented PUF. Section 4 illustrates the ChaCha20 architecture unified with the PUF. Section 5 shows the results of the unified PUF-ChaCha20 implementation. Finally, Section 6 presents the conclusion of the paper.

2. Transport Layer Security

Transport Layer Security (TLS) is used in end-to-end connections for computer networks, providing the cyber security requested by websites, as shown in Figure 1. TLS version 1.2 [34] showed an increase in the percentage of use in websites from 2016 to 2021. However, the new release of TLS was published in 2018, removing some insecure ciphers. TLS 1.3 [30] reported a five times greater amount of usage in the last year, demonstrating the relevance of the new approach. In addition, ChaCha20 is introduced in this TLS version, providing a different solution than a cipher based on AES. The new authenticated encryption with additional data (AEAD) is constructed with ChaCha20 and Poly1305 primitives for cipher and message authentication codes, respectively.

Figure 2 illustrates a typical TLS handshake procedure. The handshake describes the series of steps for exchanging information between a *client* and a *server*. The handshake sends a *client hello* to the *server*, including the TLS version, the cipher suites, and the supported client public key. Next, the server sends a *server hello* with a certificate and cipher suites selected in response to the *client hello*. The *client* verifies the certificate provided by the *server*, authenticating the owner of the domain. Finally, the *server* decrypts

the *master* key provided by the *client*, establishing a secure symmetric encryption. The data are exchanged using the AES-GCM, AES-CCM, or ChaCha20-Poly1305 defined in the cipher suites used in the hello process.

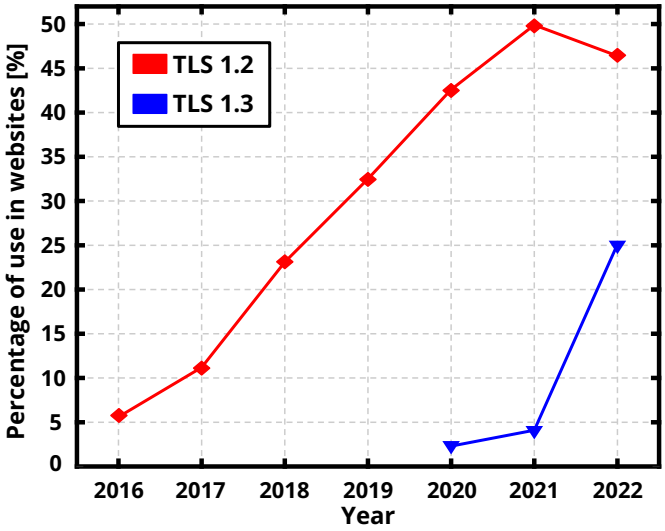

Figure 1. Percentage of use of Transport Layer Security version 1.2 and 1.3 in websites [32].

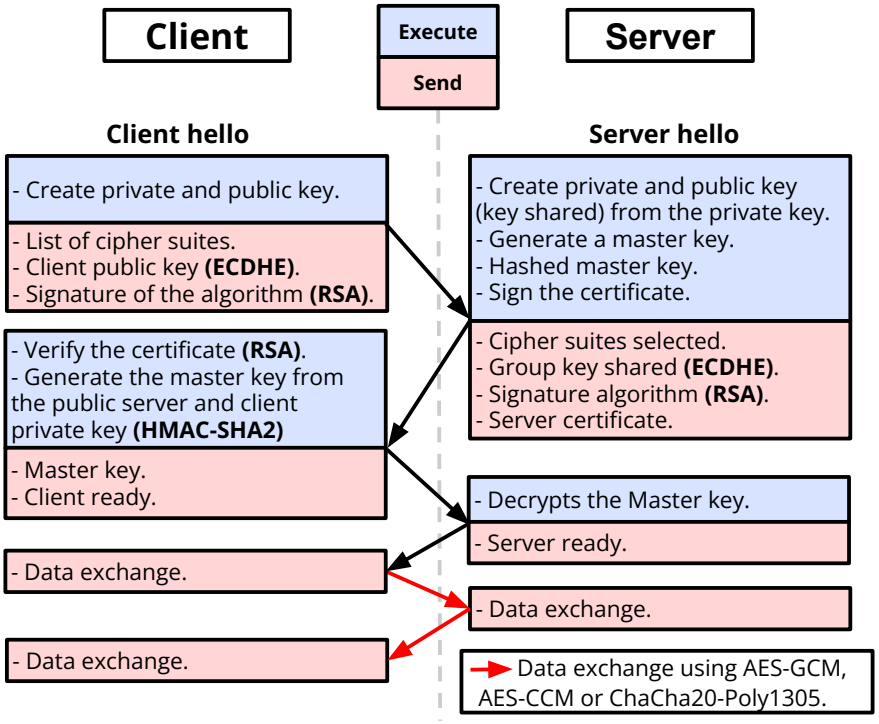

Figure 2. TransporT Layer Security 1.3 handshake protocol [32].

Figure 3 shows the ChaCha20-Poly1305 AEAD procedure. The procedure starts with the generation of the one-time key (OTK) used in the Poly1305 message authentication code (MAC) process of the AEAD. Conventionally, ChaCha20 generates an OTK using the values of the key and nonce with a counter and plaintext initialized in zero. In addition, the OTK can also be created using a linear-feedback shift register (LFSR) [35]. Next, the additional authenticated data (AAD) with an arbitrary length is introduced in the Poly1305. Then, the encryption/decryption process is applied to the *plaintext* or *ciphertext*. However, sometimes the *plaintext* or *ciphertext* in step 3 is not a full block with 512 bits, requiring

an output filter in the ChaCha20. When the encryption/decryption process is finished, a final block is introduced in Poly1305, containing the lengths of data processed and the introduced AAD. Finally, the MAC is generated.

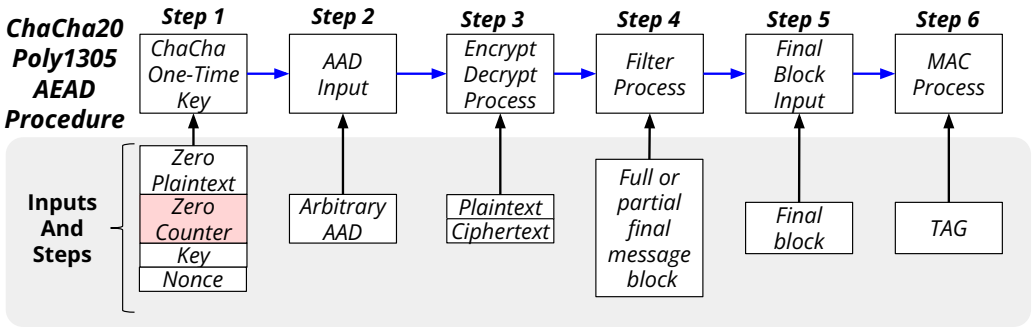

Figure 3. ChaCha20-Poly1305 AEAD procedure.

Figure 4 illustrates the typical and proposed PUF-based OTK generation. The proposed OTK generation uses a PUF unified with a crypto core, obtaining the counter's initial value. In the proposed OTK generation, the ChaCha20 takes the key, nonce, zero-initialized *plaintext* and the counter value obtained in the PUF response, mitigating the potential risks caused by the initial value change in the counter in the zero-initialized counter. After 20 rounds of ChaCha20, the first 256 bits of the *CipherText* is the OTK.

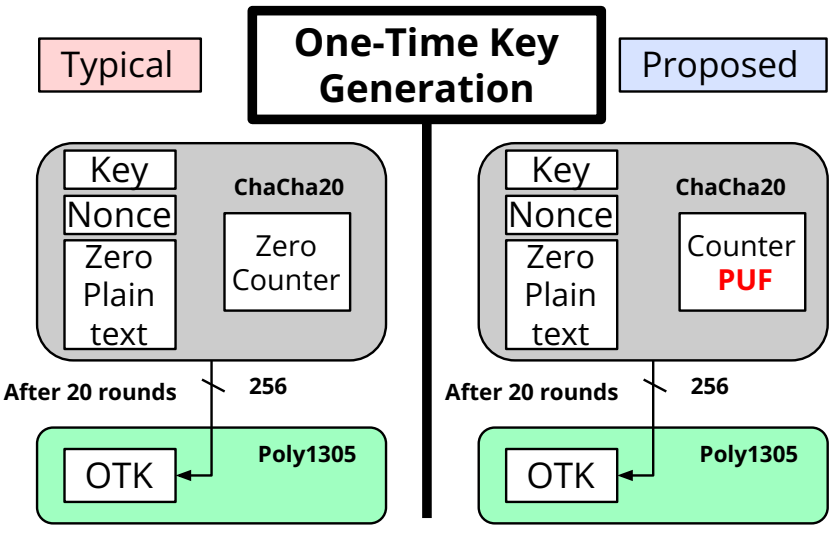

Figure 4. Block diagram of one-time key generation.

3. Physical Unclonable Function

3.1. Physical Phenomena

Figure 5 illustrates the typical physical phenomena that generate static entropy for a PUF application. Commonly, static entropy is obtained by the process variation of the implementation, which causes alterations in the frequency response in ring oscillators (ROs) [36,37] and the time of frequency collapse in multi-modal RO [38,39]. However, external circuits are necessary for collecting the static entropy. On the other hand, the dynamic entropy caused by metastability can affect the typical response of the latch, generating a random number [20–22]. Nevertheless, the process variation can generate enough static entropy in latches.

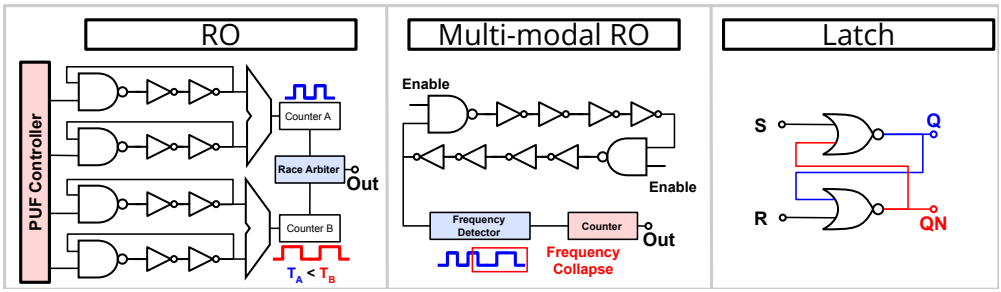

Figure 5. The PUF architectures based on RO, multi-modal RO, and latches.

Figure 6 shows the response scenarios in a metastable latch. First, the latch presents the same probability of obtaining a one and zero in the response when the latch is balanced. The balanced latches are used for a TRNG application. However, the process variations impose problems when balancing the latch, which require a calibration step [7,20,21]. On the other hand, the unbalanced latches produce a stable and unique response. Therefore, the stable and unique response in the unbalanced latches can be used for PUF applications, exploiting the process variations in the latch. In addition, the latches are used as a memory, independent of the balance. Therefore, the unbalanced and balanced latches can be combined with digital implementations, unifying the logic functions with a PUF or a TRNG application.

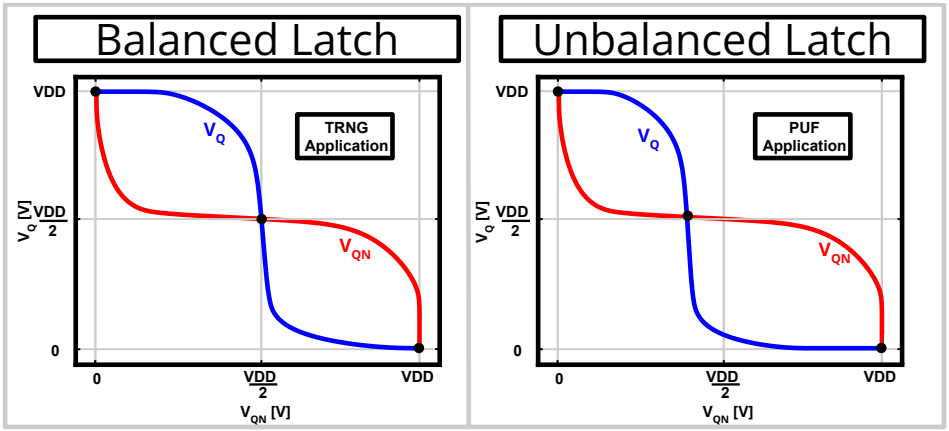

Figure 6. Metastability response in balanced and unbalanced latches.

3.2. Analytical Model

The analytical model of the PUF is based on the metastability response in latches. The metastability in latches has been studied for TRNG applications, presenting an analytical model depending on the mismatch and noise based on a loop gain model [17]. The voltage in the nodes V_Q and V_{QN} are modelled using functions depending on the opposite voltage node ($V_Q = f(V_{QN})$ and $V_{QN} = f(V_Q)$). In addition, the mismatch ($\delta_{m(Q)}$ and $\delta_{m(QN)}$) and noise ($\delta_{n(Q)}$ and $\delta_{n(QN)}$) are added to the response of each voltage node in (1).

$$\begin{aligned} V_Q &= g(V_{QN}) + \delta_{m(Q)} + \delta_{n(Q)} \\ V_{QN} &= f(V_Q) + \delta_{m(QN)} + \delta_{n(QN)} \end{aligned} \tag{1}$$

When the latch is balanced, the $\delta_{n(Q)}$ and $\delta_{n(QN)}$ generate a random response. In addition, the $\delta_{m(Q)}$ and $\delta_{m(QN)}$ impose a bias in the latch response. The undesirable effects in each node generate voltage variations in each node. When the V_Q and V_{QN} exceed the maximum voltages tolerated ($V_{(QF)}$ and $V_{(QNF)}$), the response of the latch is obtained. The result of the loop gain model approximates the response to ($V_Q = (a * V_{QN}) + b$). The coefficients a and b are calculated using the $V_{(QF)}$ and $V_{(QNF)}$ values in (2), respectively.

$$a = \frac{f(V_{QF}) - V_{QNF}}{V_{QF} - g(V_{QNF})}$$
$$b = \frac{V_{QF}V_{QNF} - f(V_{QF})g(V_{QNF})}{V_{QF} - g(V_{QNF})} \quad (2)$$

Finally, the response of the latch in a metastable response is denoted by $[(a * V_Q) + b] > [f(V_Q) - \delta_{m(QN)} - \delta_{n(QN)}]$. The latch response depends on the relationship between each node's voltage, noise, and mismatch. Therefore, a PUF implementation needs to increase the process variations represented in the mismatch and reduce the noise introduced in the latch.

3.3. Implementation

Figure 7 shows the architecture of the implemented PUF. The PUF exploits the metastability in NOR latches. The PUF architecture is designed to be unified with a crypto core, changing the operational mode of the latches between PUF and memory with the signal *mode*. Therefore, the *counter's* value can be generated in the PUF mode or stored in the crypto core mode. The PUF implementation consists of two branches of latches, and the 32-bit challenge selects which branch is used for the PUF output response. Each branch consists of 32 PUF latches. The implemented latches present an unbalanced response in all bits due to the lack of calibration and implementation strategies to improve the mismatch. The length of the challenge and response of the PUF implementation is 32 bits. In addition, the unified PUF generates an overhead of one cycle in the crypto core function.

Figure 8 illustrates the techniques applied to reduce the noise and improve the process variation to obtain unbalanced latches in the implementation. The latches are constructed with NOR cells using the LUT-6 primitive of the FPGA. In addition, the $INIT$ value of the NOR function in the LUT differs in each latch. Therefore, the mismatch of the implemented latch increases in the PUF application. Additionally, a placement blockage is implemented in the slices of the latch, reducing the noise introduced by external circuits into the latch. On the other hand, the static entropy source implemented in the latch must operate as an elementary bit memory. The entropy source is implemented in one of the internal stages of a ChaCha20 crypto core as described in Section 4. Finally, the PUF-ChaCha20 is connected to a system on chip (SoC) based on an RISC-V for testing the implementation [40].

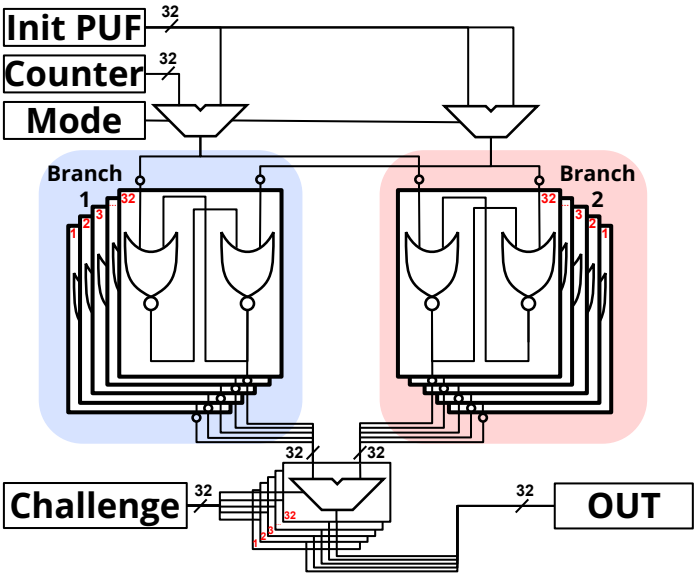

Figure 7. PUF block diagram implementation.

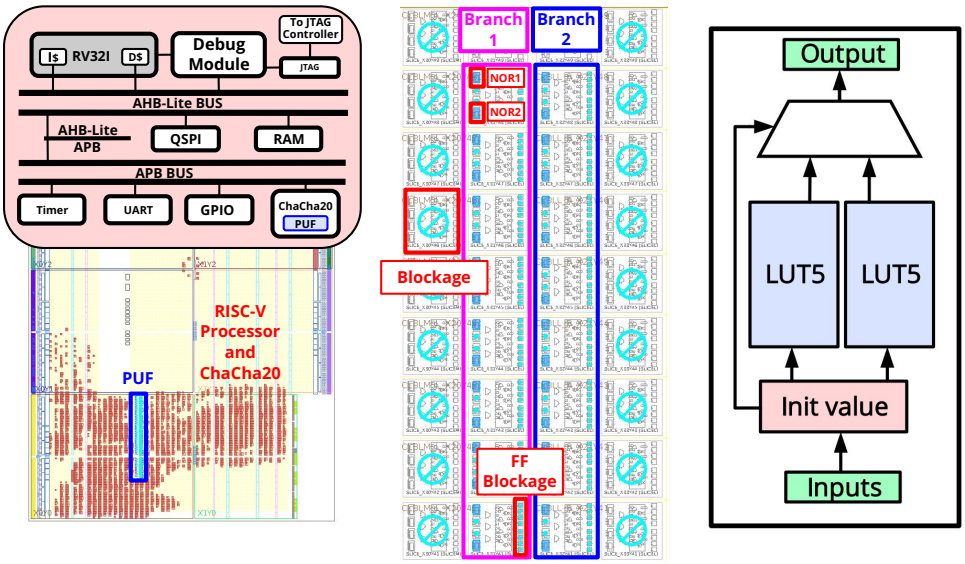

Figure 8. Strategies and constraints in the PUF implementation.

4. ChaCha20

The ChaCha20 cipher is implemented using a permutation matrix using a nonce, key, counter, and some constants [31,32]. This cryptography function is based on the Salsa20 algorithm [33]. ChaCha20 is used in TLS 1.3 to provide another solution different from the cipher based on the AES [30]. Figure 9 illustrates the architecture of the implemented ChaCha20 core. The architecture uses four quarter round (QR) operators to permute the operation regs in the block function highlighted in blue. Additionally, a finite state machine, highlighted in red, controls the permutation of the operational regs depending on the actual round (column or diagonal). Therefore, a column or diagonal round is finished in one cycle. In the last round, the values of the initial and operation regs are added to obtain the keystream. Finally, a *crypto function*, highlighted in green, applies an XOR operation between the *plain text* and the keystream. The PUF is implemented in the *initial regs*, replacing the regs with latches. The signal *mode* switches between the memory and PUF operations in the latches. In addition, a 32-bit *challenge* is used to select one of the two branches of each latch to obtain the response of the PUF.

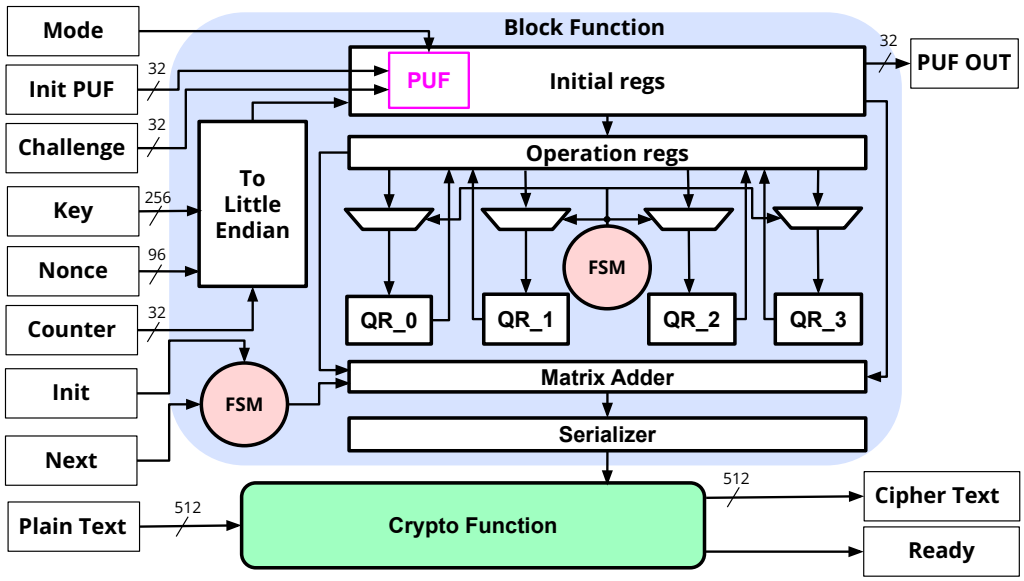

Figure 9. Architecture of the ChaCha20 crypto core [31].

Figure 10 depicts the quarter round (QR) operation implemented in the ChaCha20 crypto core. The inputs (a ,b, c, and d) each denote one stage of the operational matrix. Each stage of the initial and operational matrix consists of a 32-bit register. The results (A, B, C, and D) are stored in the same position as those used earlier in the round. The QR operator is implemented using an add, rotate, and XOR operations [41]. The rotate operation, highlighted in red, is implemented using a wire permutation to reduce the area required.

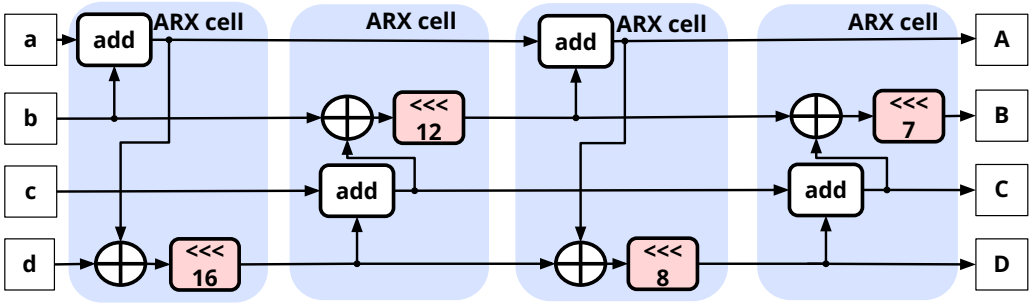

Figure 10. Implementation of the quarter round (QR) operation [31].

Figure 11 shows the organization of the initial stages, column, and diagonal rounds of the ChaCha20 matrix. The initial stages are stored in the *initial regs*, using 128 bits of constants, 256 bits of the key, 96 bits of nonce, and 32 bits of counter. However, the counter stage is implemented using the latches described in Section 3.3. ChaCha20 works in a normal function when the latches are used to store the counter's value. On the other hand, the latches in the metastable mode generate a random number depending on the process variations inside the latches. In addition, the ChaCha20 core can initialize a pseudo-key derivation using the value of the latches in metastability mode, permuting the key and the nonce used in the ChaCha20 algorithm for the ChaCha20-Poly1305 AEAD.

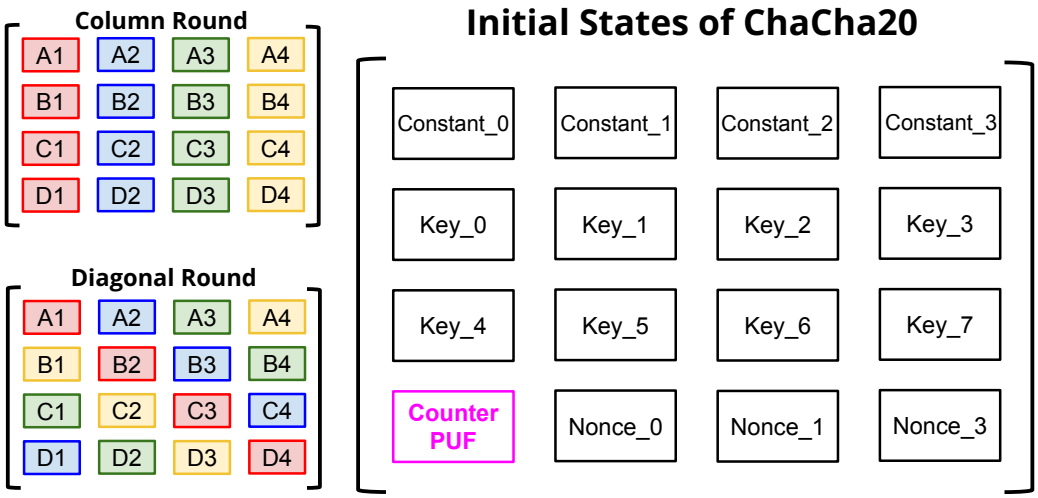

Figure 11. Structure of the column and diagonal rounds, along with the initial states of the PUF-ChaCha20 core.

5. Results

This section shows the implementation results in a Xilinx Artix-7 XC7A100TCSG324-1 FPGA. The uniqueness, uniformity, and reliability are measured in three different FPGAs. The inter- and intra-chip Hamming distances (HMD) and Hamming weights are measured in nominal conditions. The occupied resources are compared with other PUF implementations in FPGAs, which are based on different physical phenomena.

The quality of the presented PUF implementation is estimated using *uniformity*, *uniqueness*, and *reliability* [42–44]. The *uniformity* represents the average Hamming weight in the PUF response, as calculated by (3), where a is the total number of measured responses of the PUF and r_i is the number of ones in each response. The *uniqueness* denotes the PUF's capability to generate different responses between distinct devices measured using the average inter-die Hamming distance as shown in (4). There, b is the number of chips measured, and $R_{(i,j)}$ denotes the n-bit response with a particular challenge, which is the same for each chip of the two that are compared. *Reliability* indicates the ability of the PUF implementation to reproduce the same response for a particular challenge under different conditions and is measured by the intra-die Hamming distance (5). Here, c is the total number of measurements with the same challenge applied to a particular PUF. Additionally, R_f and R_i are the n-bit reference iteration responses for the same challenge, respectively. Table 1 compares the quality of the PUF responses of our implementation and the quality of the PUF responses of other implementations based on different physical phenomena. The ideal values for *uniformity*, *uniqueness*, and *reliability* are 50%, 50%, and 100%, respectively. The *uniformity*, *uniqueness*, and *reliability* values in Equations (3)–(5) are presented as normalized values.

Table 1. Summary and comparison of the PUF quality.

	This Work	**[36]**	**[45]**
Uniformity [%]	49.15	49.61	47.20
Uniqueness [%]	49.85	49.95	39.10
Reliability [%]	99.25	99.13	98.89
Topology	Unbalanced Latches	RO with Hybrid Logic	Galois RO

Figure 12 illustrates the intra- and inter-chip Hamming distances in our PUF implementation. The intra- and inter-chip Hamming distances are measured in three different FPGAs using 1000 different challenges. The challenges used in the test are randomly selected and are not repeated. The PUF exhibits an average intra-chip Hamming distance of 0.75% with a 0.2% standard variation. Additionally, the implementation exhibits an average intra-chip Hamming distance of 49.85% with a 2.2% standard variation.

$$Uniformity = \frac{1}{a}\sum_{i=1}^{a} r_i \tag{3}$$

$$Uniqueness = \frac{2}{b(b-1)}\sum_{i=1}^{b-1}\sum_{j=i+1}^{b}\frac{HMD(R_i, R_j)}{n} \tag{4}$$

$$Reliability = \frac{1}{c}\sum_{i=1}^{c}\frac{HMD(R_i, R_f)}{n} \tag{5}$$

Table 2 shows the resources occupied in the PUF-ChaCha20 implementation in an FPGA. The implementation presents a 3.1% resource overhead caused by the unified functions compared to [31]. On the other hand, the performance is compared with other ChaCha20 implementations, denoting a 0.8% performance reduction in comparison to [31] caused by the cycle necessary to configure the mode of the PUF implementation. The implementation exhibits 0.343 cycles/byte in the crypto core mode.

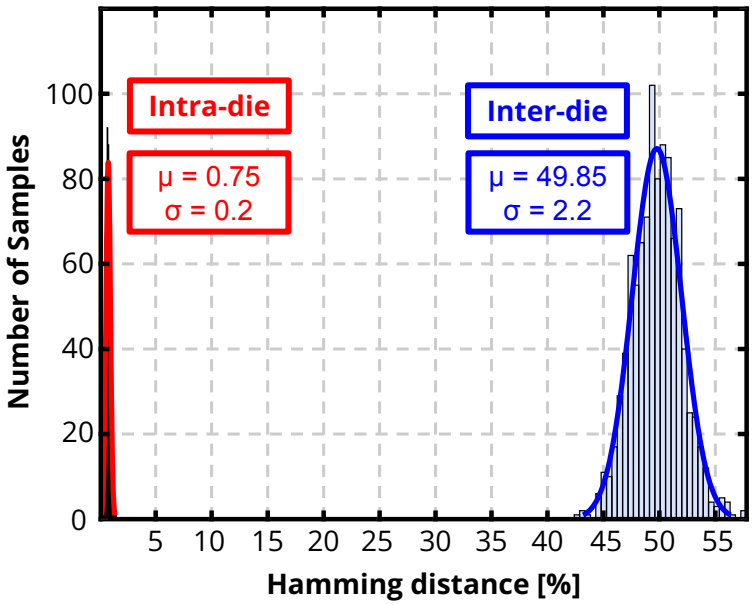

Figure 12. Inter- and intra-PUF Hamming distances in nominal conditions.

Table 2. Performance summary and comparison in the FPGA.

	Slices			Performance [Cycles/Byte]
Module	**LUT**	**FF**	**Total**	**Standalone**
This work	2416	1026	3442	0.343
[31]	2288	1058	3338	0.340
[41]	2369	2152	4521	0.530

6. Conclusions

In this paper, we present a unified PUF crypto core implementation, exploiting the metastability of the latches in an FPGA. The implemented crypto core contains the ChaCha20 cipher using a 4-QR module. The PUF is implemented in one of the stages of the ChaCha20 core using unbalanced latches. The PUF challenge is introduced to select each output bit between two branches of latches. The PUF-ChaCha20 implementation occupies 2416 LUTs and 1026 FFs, resulting in a 3.1% resource overhead. The PUF application is tested in three XC7A100TCSG324-1 Xilinx FPGAs, resulting in responses with a uniformity of 49.15%, a uniqueness of 49.85%, and a reliability of 99.25%, on average. Finally, the unified implementation can be used in OTK generation for ChaCha20-Poly1305 AEAD in TLS 1.3, generating an additional secure approach in end-to-end computer networks.

Author Contributions: Supervision, C.-K.P., C.D. and T.-T.H.; methodology, R.S., M.S. and T.-K.D.; investigation, R.S.; writing—original draft preparation, R.S.; writing—review and editing, R.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the New Energy and Industrial Technology Development Organization (NEDO) project JPNP16007.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: No new data were created or analysed in this study. Data sharing is not applicable to this article.

Acknowledgments: This paper is based on results obtained from project JPNP16007, commissioned by the New Energy and Industrial Technology Development Organization (NEDO).

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

References

1. Kumar, V.B.Y.; Chattopadhyay, A.; Haj-Yahya, J.; Mendelson, A. ITUS: A Secure RISC-V System-on-Chip. In Proceedings of the 32nd IEEE International System-on-Chip Conference (SOCC), Singapore, 3–6 September 2019; pp. 418–423.
2. Nasahl, P.; Schilling, R.; Werner, M.; Mangard, S. HECTOR-V: A Heterogeneous CPU Architecture for a Secure RISC-V Execution Environment. In Proceedings of the ACM Asia Conference on Computer and Communications Security (ASIA CCS), Virtual, 7–11 June 2021; pp. 187–199.
3. Hoang, T.-T.; Duran, C.; Serrano, R.; Sarmiento, M.; Nguyen, K.-D.; Tsukamoto, A.; Suzaki, K.; Pham, C.-K. Trusted Execution Environment Hardware by Isolated Heterogeneous Architecture for Key Scheduling. *IEEE Access* **2022**, *10*, 46014–46027. [CrossRef]
4. Taneja, S.; Rajanna, V.K.; Alioto, M. 36.1 Unified In-Memory Dynamic TRNG and Multi-Bit Static PUF Entropy Generation for Ubiquitous Hardware Security. In Proceedings of the IEEE International Solid-State Circuits Conference (ISSCC), San Francisco, CA, USA, 13–22 February 2021; Volume 64, pp. 498–500. [CrossRef]
5. Taneja, S.; Rajanna, V.K.; Alioto, M. In-Memory Unified TRNG and Multi-Bit PUF for Ubiquitous Hardware Security. *IEEE J. Solid-State Circuits* **2022**, *57*, 153–166. [CrossRef]
6. Nam, J.W.; Ahn, J.H.; Hong, J.P. Compact SRAM-Based PUF Chip Employing Body Voltage Control Technique. *IEEE Access* **2022**, *10*, 22311–22319. [CrossRef]
7. Serrano, R.; Duran, C.; Sarmiento, M.; Pham, C.K. A Unified NVRAM and TRNG in Standard CMOS Technology. *IEEE Access* **2022**, *10*, 79213–79221. [CrossRef]
8. Ray, B.; Milenković, A. True Random Number Generation Using Read Noise of Flash Memory Cells. *IEEE Trans. Electron Devices* **2018**, *65*, 963–969. [CrossRef]
9. Ardila, J.; Santamaria, J.; Florez, K.; Roa, E. A Stable Physically Unclonable Function Based on a Standard CMOS NVR. In Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS), Sevilla, Spain, 10–21 October 2020; pp. 1–4. [CrossRef]
10. Taneja, S.; Alioto, M. Fully Synthesizable Unified True Random Number Generator and Cryptographic Core. *IEEE J. Solid-State Circuits* **2021**, *56*, 3049–3061. [CrossRef]
11. Amaki, T.; Hashimoto, M.; Onoye, T. An Oscillator-based True Random Number Generator with Jitter Amplifier. In Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS), Rio de Janeiro, Brazil, 15–18 May 2011; pp. 725–728. [CrossRef]
12. Peetermans, A.; Rozic, V.; Verbauwhede, I. A Highly-Portable True Random Number Generator Based on Coherent Sampling. In Proceedings of the 29th International Conference on Field Programmable Logic and Applications (FPL), Barcelona, Spain, 9–13 September 2019; pp. 218–224. [CrossRef]
13. Chen, T.; Ma, Y.; Lin, J.; Cao, Y.; Lv, N.; Jing, J. A Lightweight Full Entropy TRNG With On-Chip Entropy Assurance. *Trans. Comput.-Aided Des. Integr. Circuits Syst.* **2021**, *40*, 2431–2444. [CrossRef]
14. Wieczorek, P.; Golofit, K. Metastability occurrence based physical unclonable functions for FPGAs. *Electron. Lett.* **2014**, *50*, 281–283. [CrossRef]
15. Yang, K.; Fick, D.; Henry, M.B.; Lee, Y.; Blaauw, D.; Sylvester, D. 16.3 A 23 Mb/s 23 pJ/b fully synthesized true-random-number generator in 28 nm and 65 nm CMOS. In Proceedings of the IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC), San Francisco, CA, USA, 9–13 February 2014; pp. 280–281. [CrossRef]
16. Yang, K.; Dong, Q.; Blaauw, D.; Sylvester, D. 14.2 A physically unclonable function with BER < 10^{-8} for robust chip authentication using oscillator collapse in 40 nm CMOS. In Proceedings of the IEEE International Solid-State Circuits Conference (ISSCC) Digest of Technical Papers, San Francisco, CA, USA, 22–26 February 2015; pp. 1–3. [CrossRef]
17. Serrano, R.; Duran, C.; Sarmiento, M.; Hoang, T.T.; Tsukamoto, A.; Suzaki, K.; Pham, C.K. A Robust and Healthy Against PVT Variations TRNG Based on Frequency Collapse. *IEEE Access* **2022**, *10*, 41852–41862. [CrossRef]
18. Park, J.; Kim, B.J.; Sim, J.Y. A PVT-Tolerant Oscillation-Collapse-Based True Random Number Generator with an Odd Number of Inverter Stages. *IEEE Trans. Circuits Syst. Ii Express Briefs* **2022**, *69*, 4058–4062. [CrossRef]
19. Serrano, R.; Duran, C.; Hoang, T.T.; Sarmiento, M.; Nguyen, K.D.; Tsukamoto, A.; Suzaki, K.; Pham, C.K. A Fully Digital True Random Number Generator With Entropy Source Based in Frequency Collapse. *IEEE Access* **2021**, *9*, 105748–105755. [CrossRef]
20. Mathew, S.K.; Srinivasan, S.; Anders, M.A.; Kaul, H.; Hsu, S.K.; Sheikh, F.; Agarwal, A.; Satpathy, S.; Krishnamurthy, R.K. 2.4 Gbps, 7 mW All-Digital PVT-Variation Tolerant True Random Number Generator for 45 nm CMOS High-Performance Microprocessors. *IEEE J. Solid-State Circ.* **2012**, *47*, 2807–2821. [CrossRef]
21. Torii, N.; Yamamoto, D.; Matsumoto, T. Evaluation of Latch-Based Physical Random Number Generator Implementation on 40 Nm ASICs. In Proceedings of the International Workshop on Trustworthy Embedded Devices (TrustED), Hofburg Palace, Vienna, 28 October 2016; pp. 23–30.

22. Tao, S.; Dubrova, E. TVL-TRNG: Sub-Microwatt True Random Number Generator Exploiting Metastability in Ternary Valued Latches. In Proceedings of the IEEE International Symposium on Multiple-Valued Logic (ISMVL), Novi Sad, Serbia, 22–24 May 2017; pp. 130–135.
23. Najm, Z.; Jap, D.; Jungk, B.; Picek, S.; Bhasin, S. On Comparing Side-channel Properties of AES and ChaCha20 on Microcontrollers. In Proceedings of the IEEE Asia Pacific Conference on Circuits and Systems (APCCAS), Chengdu, China, 26–30 October 2018; pp. 552–555. [CrossRef]
24. Saraiva, D.A.F.; Leithardt, V.R.Q.; de Paula, D.; Sales Mendes, A.; González, G.V.; Crocker, P. PRISEC: Comparison of Symmetric Key Algorithms for IoT Devices. *Sensors* **2019**, *19*, 4312. [CrossRef] [PubMed]
25. Darbar, S.; Mervin, J.; Selvakumar, D. Side Channel Leakage Assessment Strategy On Attack Resistant AES Architectures. In Proceedings of the 24th International Symposium on VLSI Design and Test (VDAT), Bhubaneswar, India, 23–25 July 2020; pp. 1–6. [CrossRef]
26. Aamir, M.; Sharma, S.; Grover, A. ChaCha20-in-Memory for Side-Channel Resistance in IoT Edge-Node Devices. *IEEE Open J. Circuits Syst.* **2021**, *2*, 833–842. [CrossRef]
27. Chou, Y.-H.; Lu, S.-L.L. A High Performance, Low Energy, Compact Masked 128-Bit AES in 22 nm CMOS Technology. In Proceedings of the International Symposium on VLSI Design, Automation and Test (VLSI-DAT), Hsinchu, Taiwan, 22–25 April 2019; pp. 1–4. [CrossRef]
28. Kumar, R.; Suresh, V.; Kar, M.; Satpathy, S.; Anders, M.A.; Kaul, H.; Agarwal, A.; Hsu, S.; Chen, G.K.; Krishnamurthy, R.K.; et al. A 4900-μm^2 839-Mb/s Side-Channel Attack-Resistant AES-128 in 14-nm CMOS With Heterogeneous Sboxes, Linear Masked MixColumns, and Dual-Rail Key Addition. *IEEE J. Solid-State Circuits* **2020**, *55*, 945–955. [CrossRef]
29. Hong, Y.-L.; Weng, Y.-K.; Huang, S.-H. Hardware Implementation for Fending off Side-Channel Attacks. In Proceedings of the IEEE International Conference on Consumer Electronics-Taiwan (ICCE-TW), Penghu, Taiwan, 15–17 September 2021; pp. 1–2. [CrossRef]
30. Rescorla, E. *The Transport Layer Security (TLS) Protocol Version 1.3*; RFC 8446; RFC Editor: Marina del Rey, CA, USA, 2018. [CrossRef]
31. Serrano, R.; Duran, C.; Hoang, T.-T.; Sarmiento, M.; Tsukamoto, A.; Suzaki, K.; Pham, C.-K. ChaCha20-Poly1305 Crypto Core Compatible with Transport Layer Security 1.3. In Proceedings of the International SoC Design Conference (ISOCC), Jeju Island, Korea, 6–9 October 2021; pp. 17–18.
32. Serrano, R.; Duran, C.; Sarmiento, M.; Pham, C.K.; Hoang, T.T. ChaCha20-Poly1305 Authenticated Encryption with Additional Data for Transport Layer Security 1.3. *Cryptography* **2022**, *6*, 30. [CrossRef]
33. Bernstein, D.J. The Salsa20 Family of Stream Ciphers. In *New Stream Cipher Designs: The eSTREAM Finalists*; Springer: Berlin/Heidelberg, Germany, 2008; pp. 84–97. [CrossRef]
34. Rescorla, E.; Dierks, T. *The Transport Layer Security (TLS) Protocol Version 1.2*; RFC 5246; RFC Editor: Marina del Rey, CA, USA, 2008. [CrossRef]
35. Nir, Y.; Langley, A. *ChaCha20 and Poly1305 for IETF Protocols*; RFC 8439; RFC Editor: Marina del Rey, CA, USA, 2018. [CrossRef]
36. Deng, D.; Hou, S.; Wang, Z.; Guo, Y. Configurable Ring Oscillator PUF Using Hybrid Logic Gates. *IEEE Access* **2020**, *8*, 161427–161437. [CrossRef]
37. Garcia-Bosque, M.; Aparicio, R.; Díez-Señorans, G.; Sánchez-Azqueta, C.; Celma, S. An analysis of the behaviour of a PUF based on ring oscillators depending on their locations. In Proceedings of the 17th Conference on Ph.D Research in Microelectronics and Electronics (PRIME), Villasimius, Italy, 12–15 June 2022; pp. 361–364. [CrossRef]
38. Zayed, A.A.; Issa, H.H.; Shehata, K.A.; Ragai, H.F. Ultra-Low Power Oscillator Collapse Physical Unclonable Function Based on FinFET. *IEEE Access* **2021**, *9*, 27696–27707. [CrossRef]
39. Park, J.; Kim, B.; Sim, J.Y. A BER-Suppressed PUF with an Amplification of Process Mismatch Effect in an Oscillator Collapse Topology. *IEEE J. Solid-State Circuits* **2022**, *57*, 2208–2219. [CrossRef]
40. Waterman, A.; Lee, Y.; Patterson, D.A.; Asanović, K. The RISC-V Instruction Set Manual, Volume I: User-Level ISA, Version 2.1. In *Technical Report UCB/EECS-2016-118*; EECS Department, University of California: Berkeley, CA, USA, 2016.
41. Pfau, J.; Reuter, M.; Harbaum, T.; Hofmann, K.; Becker, J. A Hardware Perspective on the ChaCha Ciphers: Scalable Chacha8/12/20 Implementations Ranging from 476 Slices to Bitrates of 175 Gbit/s. In Proceedings of the 32nd IEEE International System-on-Chip Conference (SOCC), Singapore, 3–6 September 2019; pp. 294–299. [CrossRef]
42. Maiti, A.; Gunreddy, V.; Schaumont, P. A Systematic Method to Evaluate and Compare the Performance of Physical Unclonable Functions. In *Embedded Systems Design with FPGAs*; Athanas, P., Pnevmatikatos, D., Sklavos, N., Eds.; Springer: New York, NY, USA, 2013; pp. 245–267. [CrossRef]
43. Gu, C.; Hanley, N.; O'neill, M. Improved Reliability of FPGA-Based PUF Identification Generator Design. *ACM Trans. Reconfigurable Technol. Syst.* **2017**, *10*, 1–23. [CrossRef]
44. Jack, M.; Máire, O. Fast DRAM PUFs on Commodity Devices. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **2020**, *39*, 3566–3576. [CrossRef]
45. Garcia-Bosque, M.; Díez-Señorans, G.; Sánchez-Azqueta, C.; Celma, S. Proposal and Analysis of a Novel Class of PUFs Based on Galois Ring Oscillators. *IEEE Access* **2020**, *8*, 157830–157839. [CrossRef]