Fuzzy Key Generator Design using ReRAM-Based Physically Unclonable Functions

Ashwija Reddy Korenda [1], Fatemeh Afghah [2], Abolfazl Razi [2], Bertrand Cambou[1], and Taylor Begay [1]

[1]School of Informatics, Computing, and Cyber Systems, Northern Arizona University, Flagstaff, Arizona.
{*ashwijakorenda, bertrand.cambou, tjb389*}*@nau.edu*

[2]Helcombe Department of Electrical and Computer Engineering, Clemson University, Clemson, SC, 29634.
{*fafghah, arazi*}*@clemson.edu*

2021 IEEE Physical Assurance and Inspection of Electronics (PAINE) | 978-1-6654-1010-6/21/$31.00 ©2021 IEEE | DOI: 10.1109/PAINE54418.2021.9707714

***Abstract*—Physical unclonable functions (PUFs) are used to create unique device identifiers from their inherent fabrication variability. Unstable readings and variation of the PUF response over time are key issues that limit the applicability of PUFs in real-world systems. In this project, we developed a fuzzy extractor (FE) to generate robust cryptographic keys from ReRAM-based PUFs. We tested the efficiency of the proposed FE using BCH and Polar error correction codes. We use ReRAM-based PUFs operating in pre-forming range to generate binary cryptographic keys at ultra-low power with an objective of tamper sensitivity. We investigate the performance of the proposed FE with real data using the reading of the resistance of pre-formed ReRAM cells under various noise conditions. The results show a bit error rate (BER) in the range of 10^{-5} for the Polar-codes based method when 10% of the ReRAM cell array is erroneous at Signal to Noise Ratio (SNR) of 20dB.**

This error rate is achieved by using helper data length of 512 bits for a 256 bit cryptographic key. Our method uses a 2:1 ratio for helper data and key, much lower than the majority of previously reported methods. This property makes our method more robust against helper data attacks.[1]

***Keywords*— Physically unclonable functions, IoT security, hardware-based authentication, ReRAM technology, Fuzzy extractors.**

I. Introduction

PUFs have been widely used for authentication and key generation at costs much lower than the conventional approach of producing secret keys by a server and distributing them among IoT devices [1]. Using PUF-based key generation in IoT devices allows securing information exchange between the low-powered IoT devices while not imposing storage overhead on their limited memory. One of the advantages of using PUFs is their ability to generate secret session keys without the need to store the keys in their embedded memory. Not storing the passwords in the memory will mitigate attacks involving key extraction from non-volatile memory. PUFs are also known to be robust against man-in-the middle attacks since these attacks usually involve cloning the secret keys. This is not possible when using a PUF-based cryptographic key generation.

As PUFs are based on the physical characteristics of a device, they are prone to environmental changes such as temperature, humidity variations and background noise. The response of an electronic PUF can even vary over time by intensive use, known as the aging factor. When using a PUF for authentication, a certain amount of error is usually acceptable which accounts for the false authentication rate (FAR) and false rejection rate (FRR) of a device. But, when a PUF is used for key generation, error is not tolerated noting that, once a message has been encrypted using a particular key, it will require the same exact key to be decrypted. Even a single bit error will not allow the proper decryption of the message, thereby leading to information loss between the devices.

Error correction codes (ECCs) are generally used in communication systems to correct errors in the transmitted messages due to channel noise, interference, fading, Doppler effect, imperfect synchronization, etc. These ECC methods cannot be used directly in PUF-based protocols because the PUF errors are induced in the original input message itself, and the error-free version is not available in the first place. When the ECCs are applied directly to PUF-based protocols, we will only multiply the errors already present in the PUF response. Therefore, specialized protocols are required to handle PUF errors. Fuzzy Extractors (FEs) are protocols used to correct input errors using the concepts of ECC in their Secure Sketch phase. FEs are usually designed in such a way that they can encapsulate all the PUF errors into another vector, which can be passed into the ECC decoder to correct the errors.

In this paper, we briefly explored the current trends in developing FEs, along with reviewing the essentials of the ECCs and their decoding schemes utilized for testing (e.g., successive cancellation, belief propagation). FEs were first proposed in [2] to extract cryptographic keys from noisy biometric data to allow reproduction of the key from any noisy input data. In FE structures, we encapsulate PUF errors into a vector to be fed into an ECC decoder, allowing the recovery of the original response from the noisy version.

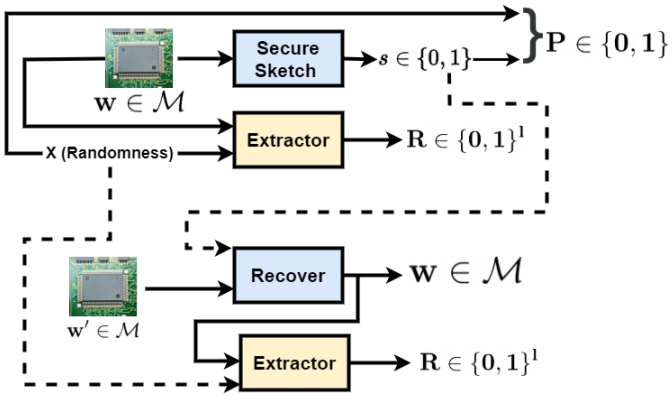


Fig. 1: Construction of a fuzzy extractor using a secure sketch.

Several methods have been used in the past to generate reliable and reproducible cryptographic keys using different types of FEs. Some of the remarkable methods include [3]–[14]. In [4], a key generation scheme was proposed which employs BCH codes in its construction. In this scheme, the key was extracted by XORing the PUF data with

[1]This material is based upon the work supported by the National Science Foundation under Grant No. 1827753.

a random string produced by a random number generator, which is made publicly available. The protocol was simulated in the MATLAB environment to check the reproducibility of the PUF data. The FE structure employs a BCH encoder and decoder to recover PUF data using noisy PUF and helper data. This scheme, when given a noisy PUF with a flipping probability of 0.15, produced a key at a bit error rate (BER) of 0.1174% from the key stored in the server. This scheme may be acceptable for authentication scenarios, where a certain error margin is tolerated. In key generation schemes, where the key is used for encrypting and decrypting data, the key error has to be zero.

In [15], a FE structure based on Polar codes for SRAM PUFs was proposed. This work utilized a complex Hash-Aided SC decoder to ensure that the key was reproducible. The results showed that the key was reproducible with a failure probability of 10^{-9}. This method utilized relatively long helper data (896 bits of Helper data for a 128 bit key).

In this paper, we propose a FE to correct the PUF reading errors with much shorter helper data. We tested our FE with real ReRAM data extracted using a ReRAM PUF protocol proposed in [16]. This protocol is briefly summarized in section II-E. Different noise conditions like burst error and random cell error were applied to the real ReRAM cell data to simulate different noise conditions.

A. Contributions

The key contributions of this paper can be summarized as follows:

- We propose a novel FE structure that allows regeneration of cryptographic keys from a PUF while utilizing 512 helper data bits for a 256 bit cryptographic key. The efficiency of this model is tested with BCH and Polar ECCs.
- We utilize and tested pre-formed ReRAM-based PUFs with multiple-level ultra-low power current readings as a robust PUF technology against side-channel attacks.
- We implement the ReRAM-based protocol integrated with the proposed FE to test the efficiency of the FE.
- The ReRAM-based PUF protocol generates very stable cryptographic keys with errors in the range of 10 parts per million. In order to ensure that the ReRAM protocol will regenerate robust cryptographic keys even in extreme conditions, we use this protocol in conjunction with the proposed FE.
- This protocol has been tested with resistances from the pre-formed ReRAM cell array. To simulate the error that might occur in extreme conditions, we use Additive White Gaussian Noise (AWGN) noise for testing.

II. Background

This section provides a brief background on the concepts on which the proposed protocol relies.

A. BCH Codes

BCH codes, invented by Bose, Chaudhuri, and Hocquenghem, are a variant of cyclic ECC codes constructed over Galois field. The design procedure ensures that up to a certain number of errors can be recovered. The BCH codes are invented in the late sixties and since then have been used in many communication and data storage applications. These codes use syndrome-based decoders, which allow simple hardware design appropriate for low-power devices.

B. Polar Codes

Proposed in 2009 by Erdal Arikan, polar codes were the first deterministic construction of ECC with low encoding and decoding complexity, which achieved Shannon capacity for binary discrete-memoryless channels (B-DMC) [17]. Polar codes are a type of linear block ECCs whose construction is based on channel polarization. The key idea is using multiple recursive concatenations of short kernel codes, which transforms the N copies of a physical channel into N virtual outer channels whose capacity gets close either to 1 or 0, if N is sufficiently large. The sum capacity of all of the virtual channels before and after the recursive concatenation is constant. Polar codes use this channel polarization property. The information in polar codes is sent only through the channels whose channel capacity is 1 (unfrozen bits). This approach helps us use the high-capacity channels for error recovery in the PUF response and discard the bits over low-capacity channels (frozen bits).

C. ReRAM PUF

The ReRAM PUFs that we designed for this work use *memristor* cells that operate at very low electrical currents. The small currents injected to allow the formation of ephemeral conductive paths in each cell. The resistance values randomly vary cell-to-cell, at different levels of injected current [18]. These resistances are used to extract the ReRAM PUF "fingerprint". These PUFs are not prone to side channel analysis attacks owing to the very low power dissipated from the device. The protocol to extract a fingerprint using ReRAM memory is described in section II-E.

Pre-Formed ReRAMs: An important physical property that we use in our PUFs is the resistance of the ReRAM cells measured on pristine cells at very low injected currents, in the so-called "pre-forming" range. The currents are so low that no visible drifts in the resistance values are observed, which is desirable for the reliability of the PUFs. Other important properties include:

- Extremely low variations of the resistance values of the same cells when subjected to repetitive measurements. This is important to minimize the intra-PUF BER. Typical intra-PUF relative standard variation of the resistance was measured in the 1% range.
- High variations of the cell-to-cell resistance values. This is important to maximize entropy, leading to the generation of high-entropy cryptographic keys without BER. Typical cell-to-cell relative standard variation of the resistances was measured in the 25% range.
- The optimal range of the injected current in which the measurements are highly reproducible and does not disturb the cells is 50 nA to 1μA. Below 50 nA, the accuracy of the measurements is questionable; above 5μA, we observed that the partial forming of the cells results in a permanent lowering of the resistances.

D. Device Specification

Al/AlOx/W devices were used in this study with a device size of 180 nanometer (nm). The bottom electrode, Tungsten ('W'), was deposited by sputtering, the switching layer ('AlOx') was deposited by atomic layer deposition (ALD), and the top electrode, Aluminum ('Al'), was placed by reactive sputtering. Both the switching layer and top electrode are Al-based materials, this design was intentional for protection against invasive attacks: when the device is switched 'ON' by a conductive filament, it is not possible to identify the nm-wide Al filament in the switching layer by Transmission Electron Microscopy (TEM) analysis. Furthermore, an attacker cannot tell whether Al material is from the top electrode or originally in the switching layer. The equipment used to characterize the devices was executed by Keysight's B1500A Semiconductor Analyzer using a High-Resolution Source Measure Unit (HRSMU) card. The devices underwent forced current injections by current sweeps from 50 to 800nA in 50nA increments providing 16 unique resistances per cell. The devices are read in reverse bias to reduce drifting effects of ion migration. Each device was consecutively read for 50 cycles for the 16 different currents to understand the noise level of each cell at different currents. An example of resistance for 2 different cells at currents measured over 10 cycles is shown in table I.

E. ReRAM protocol

1) Server Level: The ReRAM protocol used to extract the PUF output has been proposed in our previous work in [16]. Initially, a

	Cell 1									
Current	**Cycles**									
100 nA	1798000	1803200	1800000	1794800	1799200	1797200	1809600	1791200	1802800	1795200
400 nA	1103700	1104800	1102700	1103500	1102100	1102400	1103500	1101200	1102000	1102700
	Cell 2									
Current	**Cycles**									
100 nA	1938000	1942000	1954800	1952000	1954800	1957600	1950000	1962800	1948400	1941200
400 nA	1180300	1187600	1187700	1196200	1190100	1179900	1183000	1178800	1192600	1187600

TABLE I: Sample Data extracted from 2 different ReRAM Cells over 2 currents(100nA and 400nA) over 10 cycles.

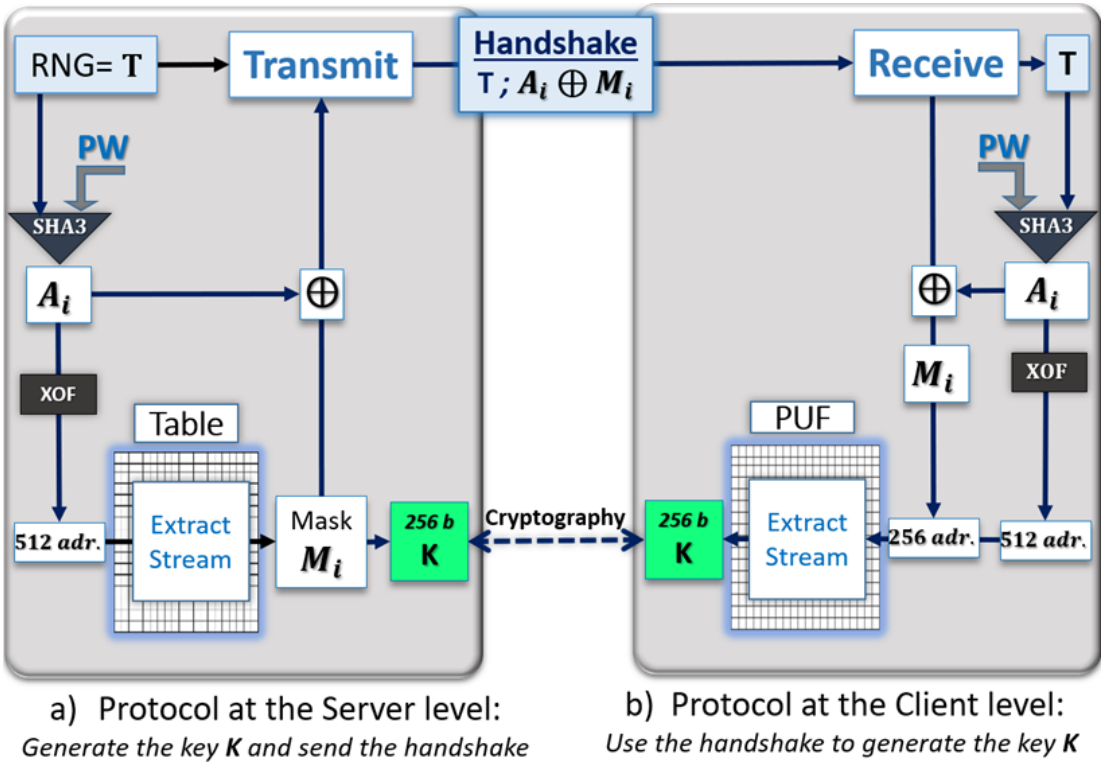


Fig. 2: The ReRAM key generation protocol.

random number T and password are XORed and hashed to extract a message digest A. The last 24 bits of the message digest are selected to define the current level to be injected into the PUF. A is then sent to an eXtendable Output Function (XOF), which is needed to obtain enough bits from A. For example, if we have a 1 Mbit array, we need 10 bits to find the X-axis and 10 bits to find the Y-axis of a particular cell. Therefore, to select 512 cells, we need a 10,240-bit long stream out of the XOF.

The median of the current readings of the cell array is calculated and called *reference median* (R_{med}). For each cell, the resistance value R_i under the injected current is compared to the R_{med}. A vector C_i is generated such that with element values '0' and '1', respectively, if $R_{med} > R_i$ or $R_{med} < R_i$.

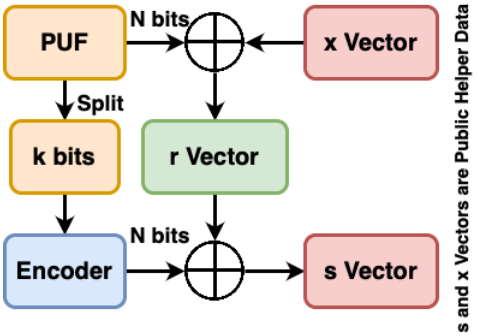
In order to ensure that the cryptographic key is stable, equal number of the most stable 0s and 1s in C are selected as the key. A mask M is created to help allow the client side to select same cryptographic key as the server side. The address of the reference cell is appended to the mask M in order to find the R_{med} on the client end. The output of XOR of M and A is sent to the client using handshake.

2) Client Level: At the client end, we use the information sent through handshake to extract A and M. We use M to extract the address of the reference cell to extract the R'_{med}. Using M, we only read values of the cells, which are stable (cells represented by 0 in the M vector), and compare them with the reference median to calculate our cryptographic key. The ReRAM protocol at the server and client level is summarized in Figure 2.

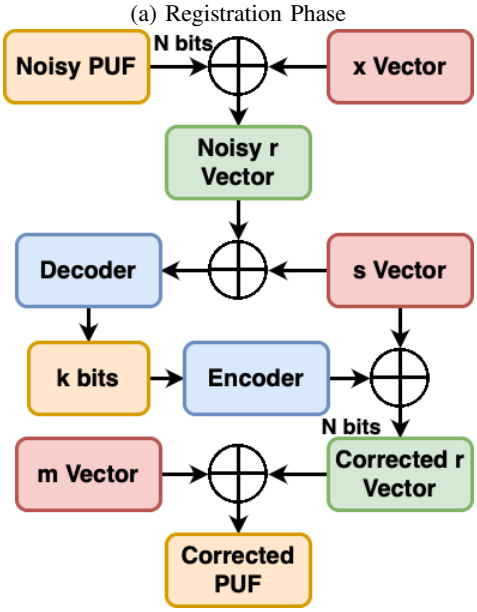
III. Proposed Fuzzy Extractor Model

We developed a basic FE protocol as shown in Figure 3. We initially use the entire PUF data (N bits) and XOR it with a random number x to calculate the vector r Part of the PUF response with the size of k bits is encoded with the ECC to extract the encoded version (N bits). This encoded version is XORed with the vector r to calculate the vector s. This protocol uses the vectors s and x as helper data.

In the authentication phase, when we have access to the Helper data and the Noisy PUF data, we XOR x with the Noisy PUF to calculate the Noisy vector r. This vector encapsulates all the Noisy bits from the Noisy PUF data. It is then XORed with vector s, and the resulting output is sent to the ECC decoder to remove the noise and calculate the corrected part of the PUF response/output (k bits). The corrected vector k is encoded with the ECC encoder whose output is XORed with s to calculate the corrected version of r. This r is then XORed with x to calculate the original PUF response, removing all the errors from the Noisy PUF.



(a) Registration Phase



(b) Authorization phase

Fig. 3: Proposed FE Protocol

IV. Preliminary Results and Analysis

This protocol has been tested using Polar codes and compared with BCH codes at different lengths of K vectors. Varying the length of K will change the error correction efficiency of the code. For testing, we utilized Polar codes of codeword length $N = 256$ with multiple message lengths $K = 32, 64, 128$ as well as the BCH codes of codeword length $N = 255$ with data-bit lengths $K = 37,\ 63$ and 131. The BER comparison between the Polar and BCH codes is shown in Figure 4, which shows that the proposed architecture equipped with Polar codes has a better error correction capacity than the BCH codes under similar noise conditions. The average BER of the polar code of length $N = 256$ is 0.043% while the BER of the BCH codes of length $N = 255$ is 0.055% at SNR: -10 dB. This shows about 20% improvement for the error recovery rate of the utilized Polar codes compared to the BCH codes with similar or longer codewords.

These results will be used to check the stability of the proposed FE architecture. Once the physical implementation of the architecture using a System on Chip (SoC) is finalized, we will work on developing the FE structure for Ternary data. For this purpose, we will initially analyze the ReRAM data we have and extract any feature which might affect the decoder structure.

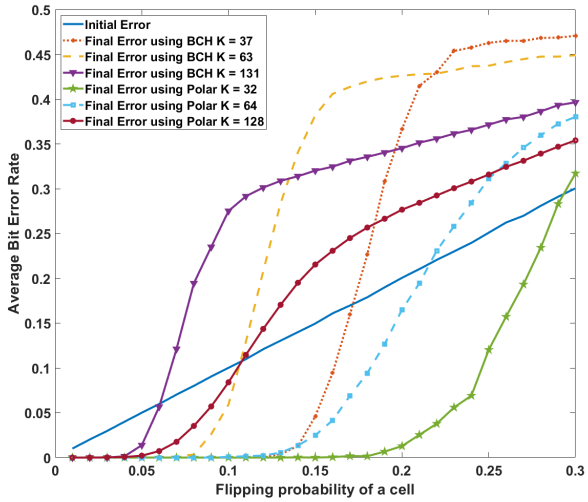


Fig. 4: Comparison of BER of the FE using BCH and Polar Codes

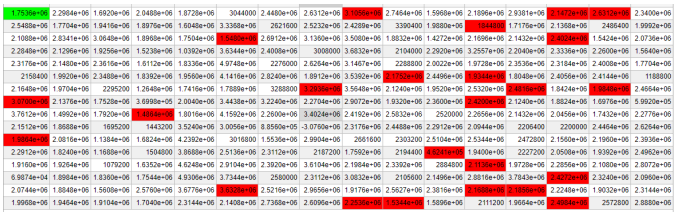


Fig. 5: Example of Random Errors in ReRAM cell array.

V. Experiments and Results

After investigating the efficiency of the proposed FE architecture using simulated data, we applied the developed FE to the real ReRAM data (see section **??**). As we have resistance data from 309 cells of a ReRAM array at 16 different currents, we assume a $16 \times 16 = 256$ ReRAM array that allows visualization of a square ReRAM array. We utilize the ReRAM protocol described in section II-E to extract PUF data, applied the proposed FE, and the results were observed under different noise conditions. These tests have been conducted 1000 times, and the BER results are obtained by averaging over all executions of the algorithm.

A. Test with random error in Real ReRAM data

In our simulations, we assume that 10% of the cells are erroneous due to the random errors as well as the inherent PUF response variations. For simulation, 25 random cells from the 256 ReRAM cell array were randomly selected, and an AWGN is added to the readings of these cells to simulate the PUF response variation under extreme conditions. These noisy cells will be used in the authentication phase (at the client level of the protocol). We applied the ReRAM protocol coupled with the FE to this data. In the registration phase (at the server end), the initial PUF response is used to create the Helper data and the cryptographic key. In the authentication phase, we applied a 10% random cell error in the PUF. This can be seen as the red cells in Figure 5. This erroneous array is used in the authentication phase to extract the cryptographic key. We explored the use of different lengths of the k vector in both Polar and BCH codes. The increase in the length of the k vector was expected to decrease the correction capability of the utilized ECC. We use the Helper Data

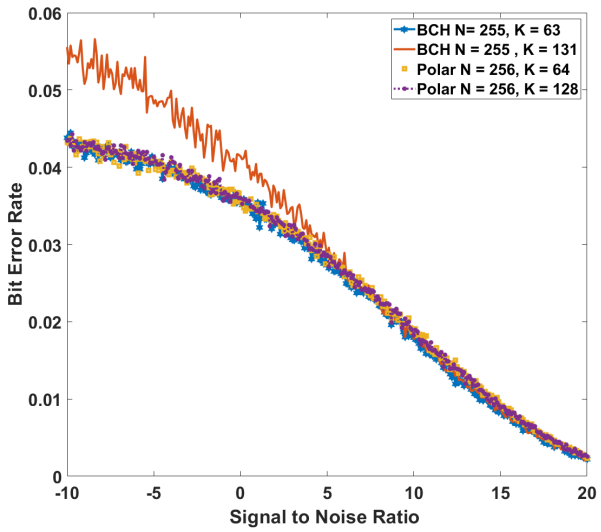


Fig. 6: Performance of BCH and Polar codes with random error at different lengths of the vector k.

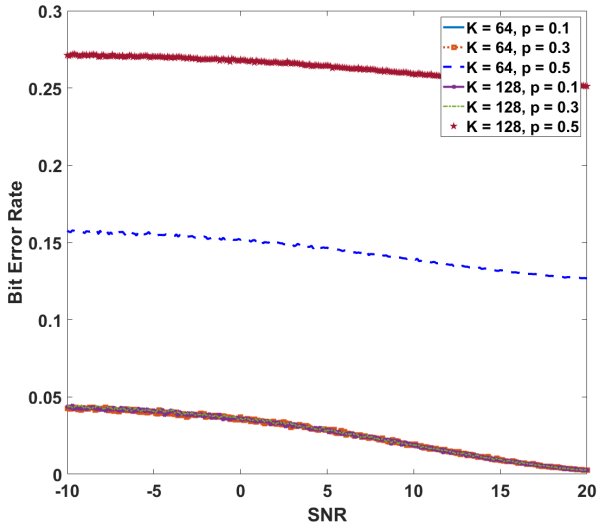


Fig. 7: Performance of Polar codes $K = 128, 64$ with random error at different values of channel parameter p.

in the authentication phase protocol to regenerate the cryptographic key.

Figure 6 shows the effect of the SNR of the AWGN on the BER percentage of the ReRAM protocol-based FE. For this test, we randomly selected 10% of the cells in the ReRAM array to be erroneous at different lengths of the vector k using BCH and Polar codes. We can observe a slight decrease in the BER of the BCH codes by decreasing the length of the input vector k, especially at low SNR values. However, this effect is not observed or is negligible when Polar Codes are utilized, which can be considered as another advantage for using Polar codes.

Figure 7 shows the effect of the channel crossover (i.e., bit flipping) parameter p used in the calculation of the Bhattacharya parameter, which helps determine the capacity of the channels in a Polar code-based FE. The results show that when $p = 0.1, 0.3$ are selected, the BER is very low and almost in the same range for different SNR

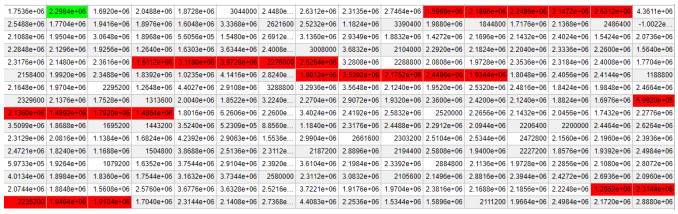


Fig. 8: Example of Burst Error in the ReRAM cell array.

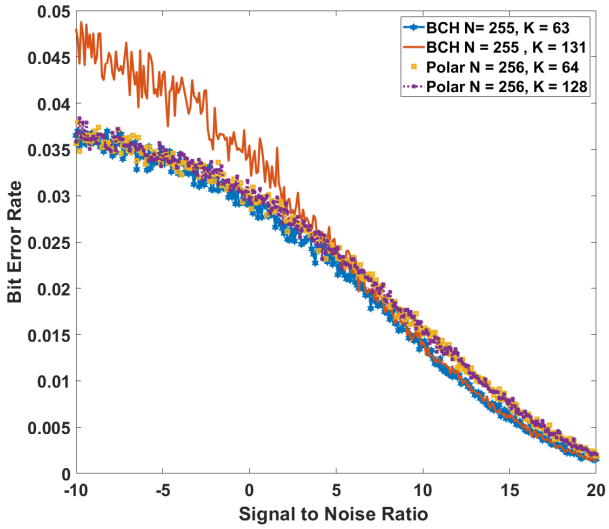


Fig. 9: Performance of BCH and Polar codes with Burst error at different lengths of the k vector

ratios. When the value of p increases beyond 0.3, the performance of the code in terms of BER decreases by more than 50%. This increase in the error rate could be due to the fact that at higher p values, the channel assumes a higher percentage of error in the bits, and as the error is lower than the expected range, it over corrects the data thereby causing more errors.

B. Burst Error

To simulate burst errors, we assume that 10% of the neighbour cells are in error. More specifically, we randomly select 5 start positions with 5 adjacent cell positions in the 256 ReRAM cell array to be erroneous (Burst error). These noisy burst error cells are used in the authentication phase of the protocol. Similar to the previous scenario (random error), we apply the same protocol to extract the key. The Burst error can be seen by the red cells in Figure 8.

Figure 9 presents the effect of SNR on the achieved BER when the length of the vector k changes in BCH and Polar codes. Similar to the random error scenario, there is a slight decline in the BER when decreasing the length of k in the case of Burst errors compared to Random errors in both the coding schemes. This could be due to the way the ReRAM protocol selects the cell addresses, since there is a low probability that all the adjacent cells which are part of the Burst error are picked in generating the cryptographic key. Similar to the random error, when the BER percentage is very low, the effect of the length of vector k is quite negligible in Polar codes with burst error, while there is a slight decrease in the BER at low SNRs when using BCH codes.

Figure 10 shows the effect of the channel parameter p, used in calculating the Bhattacharya parameter, at $K_{vector} = 64$ and 128.

TABLE II: Comparison of a few error correction schemes with our proposed model

Fuzzy Extractor scheme	ECC Utilized	PUF used	Length of Key	Length of Helper Data
BCH Repetition Code [19]	Ring Oscillator PUF	Optical PUF	128	2052
Reed Muller Generalized multiple concatenated (GMC) Coding [20]	**Inner code:** Soft decision maximum likelihood (SDML) decoder. **Outer Code:** GMC Reed Muller Decoder	SRAM PUF	128	13952
FE based on Polar Codes [15]	Polar(N = 1024)	SRAM PUF	128	896
Proposed FE	BCH (N = 255)	ReRAM PUF Operating in Pre-forming Range	255	510
Proposed FE	Polar (N = 256)	ReRAM PUF Operating in Pre-forming Range	256	512

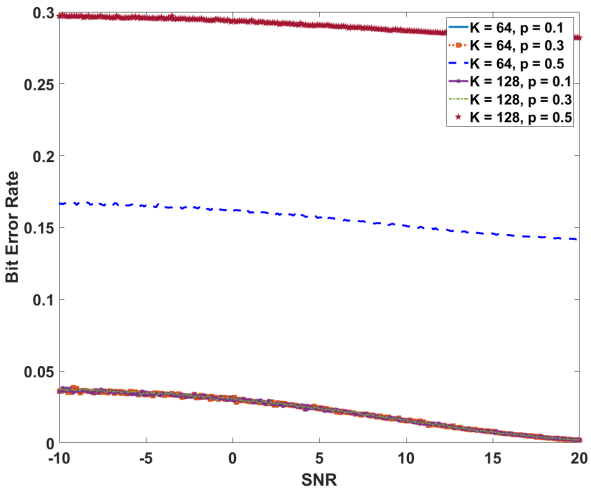


Fig. 10: Performance of Polar codes $K = 128, 64$ with Burst error for different channel parameter p.

Similar to the random error case, the results show that when p is low, the BER is very low and almost in the same range for different SNR values. When the value of p increases beyond 0.3, the achieved BER decreases. Table II compares the characteristics of the proposed method with some previously reported methods, in terms of the key length, the codeword length, the length of helper data, the utilized memory technology, and the type of the employed ECC. A key observation is that we achieved superior ECC performance with much lower helper data length and shorter codewords using ReRAM PUFs operating in Pre-forming Range.

VI. Conclusion and Discussion

We developed a fuzzy extractor using BCH and Polar codes and tested its efficiency using real data obtained from ultra-low-power pre-forming ReRAM cells. The results were generated based on a random error and 10% burst error in the ReRAM array. Our technical inspection revealed that the actual BER in cell arrays is in the orders of 10 parts per million, which is much lower than our simulated error. This gives confidence for using our method even with shorter helper data.

The proposed Fuzzy Extractor allows key extraction with an error rate of $5 * 10^{-5}$ with a random cell error percentage of 10. The performance of the fuzzy extractor was analyzed with BCH and Polar codes under different noise conditions. Overall, the proposed error rate is much lower than previously reported methods at similar conditions and using shorter helper data, which makes our method robust against helper data attacks. Also, using the ultra-low-power technology substantially mitigates the PUF's vulnerability against side-channel and allows the PUF device to be tamper resistant.

References

[1] A. Shamsoshoara, A. Korenda, F. Afghah, and S. Zeadally, "A survey on physical unclonable function (puf)-based security solutions for internet of things," Computer Networks, vol. 183, p. 107593, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1389128620312275

[2] Y. Dodis, L. Reyzin, and A. Smith, "Fuzzy extractors: How to generate strong keys from biometrics and other noisy data," in International conference on the theory and applications of cryptographic techniques. Springer, 2004, pp. 523–540.

[3] G. E. Suh and S. Devadas, "Physical unclonable functions for device authentication and secret key generation," in Proceedings of the 44th annual design automation conference. ACM, 2007, pp. 9–14.

[4] H. Kang, Y. Hori, T. Katashita, M. Hagiwara, and K. Iwamura, "Performance analysis for puf data using fuzzy extractor," in Ubiquitous Information Technologies and Applications. Springer, 2014, pp. 277–284.

[5] K. Kursawe, A.-R. Sadeghi, D. Schellekens, B. Skoric, and P. Tuyls, "Reconfigurable physical unclonable functions-enabling technology for tamper-resistant storage," in Hardware-Oriented Security and Trust, 2009. HOST'09. IEEE International Workshop on. IEEE, 2009, pp. 22–29.

[6] T. Ziola, Z. Paral, S. Devadas, G. E. Suh, and V. Khandelwal, "Authentication with physical unclonable functions," Jul. 15 2014, uS Patent 8,782,396.

[7] B. Škorić, P. Tuyls, and W. Ophey, "Robust key extraction from physical uncloneable functions," in International Conference on Applied Cryptography and Network Security. Springer, 2005, pp. 407–422.

[8] D. Lim, J. W. Lee, B. Gassend, G. E. Suh, M. Van Dijk, and S. Devadas, "Extracting secret keys from integrated circuits," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 13, no. 10, pp. 1200–1205, 2005.

[9] H. Kang, Y. Hori, T. Katashita, and M. Hagiwara, "The implementation of fuzzy extractor is not hard to do: An approach using puf data," in Proceedings of the 30th Symposium on Cryptography and Information Security, Kyoto, Japan, 2013, pp. 22–25.

[10] J. Delvaux, D. Gu, I. Verbauwhede, M. Hiller, and M.-D. M. Yu, "Efficient fuzzy extraction of puf-induced secrets: Theory and applications," in International Conference on Cryptographic Hardware and Embedded Systems. Springer, 2016, pp. 412–431.

[11] M. Taniguchi, M. Shiozaki, H. Kubo, and T. Fujino, "A stable key generation from puf responses with a fuzzy extractor for cryptographic authentications," in Consumer Electronics (GCCE), 2013 IEEE 2nd Global Conference on. IEEE, 2013, pp. 525–527.

[12] N. Price, How to generate repeatable keys using physical unclonable functions: Correcting PUF errors with iteratively broadening and prioritized search. University of Maryland, Baltimore County, 2014.

[13] A. R. Korenda, F. Afghah, and B. Cambou, "A secret key generation scheme for internet of things using ternary-states reram-based physical unclonable functions," in 2018 14th International Wireless Communications Mobile Computing Conference (IWCMC), 2018, pp. 1261–1266.

[14] A. R. Korenda, F. Afghah, B. Cambou, and C. Philabaum, "A proof of concept sram-based physically unclonable function (puf) key generation mechanism for iot devices," in 2019 16th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON), 2019, pp. 1–8.

[15] B. Chen, T. Ignatenko, F. M. Willems, R. Maes, E. van der Sluis, and G. Selimis, "High-rate error correction schemes for sram-pufs based on polar codes," arXiv preprint arXiv:1701.07320, 2017.
[16] B. Cambou, "Fast cryptographic key generator from memristor based physically unclonable computed from it." U.S. provisional patent application serial no.63/004,4938; NAU Case no. 220-031.
[17] E. Arikan, "Channel polarization: A method for constructing capacity-achieving codes for symmetric binary-input memoryless channels," IEEE Transactions on information Theory, vol. 55, no. 7, pp. 3051–3073, 2009.
[18] B. Cambou, D. Hély, and S. Assiri, "Cryptography with analog scheme using memristors," ACM Journal on Emerging Technologies in Computing Systems (JETC), vol. 16, no. 4, pp. 1–30, 2020.
[19] R. Maes, A. Van Herrewege, and I. Verbauwhede, "Pufky: A fully functional puf-based cryptographic key generator," in International Workshop on Cryptographic Hardware and Embedded Systems. Springer, 2012, pp. 302–319.
[20] R. Maes, P. Tuyls, and I. Verbauwhede, "Low-overhead implementation of a soft decision helper data algorithm for sram pufs," in International Workshop on Cryptographic Hardware and Embedded Systems. Springer, 2009, pp. 332–347.