Authentication in High Noise Environments using PUF-Based Parallel Probabilistic Searches

Brian Donnelly
School of Informatics, Computing and Cyber Systems
Northern Arizona University
Flagstaff, Arizona
brian.donnelly@nau.edu

Michael Gowanlock
School of Informatics, Computing and Cyber Systems
Northern Arizona University
Flagstaff, Arizona
michael.gowanlock@nau.edu

***Abstract*—Enabling secure communication in noisy environments is a major challenge. In these environments, the outputs of cryptography algorithms undergo error where several bits change states and since these algorithms cannot tolerate any error, authenticating and securing communication between parties is disrupted. We propose a noise-resistant public key infrastructure protocol that employs physical unclonable functions (PUFs). PUFs act as a unique fingerprint for each device in a network; however, their state may drift over time due to fluctuations in temperature and other factors. Using a PUF requires a search to identify flipped bits which is conducted on a secure server that has the benefit of removing error correction on low-powered client devices. We exploit the probabilistic nature of PUF bit error rates (BERs) and use this information to aid in the search process that resolves the noise imparted by the environment. We show that using a 256-bit PUF-generated seed (a PUF response) our protocol is robust to a PUF BER of $\approx$11% (or 30 of 256 bits) and a transmission bit error rate (TBER) of 30%. In this scenario, on average the authentication mechanism on a secure server requires $\lesssim$5 s. We also show results for higher PUF BERs which have a $<$100% authentication success rate which indicates the upper limit on the PUF BER tolerance of our protocol.**

***Index Terms*—AES, Communication, Hashing, Parallel Computing, Response-Based Cryptography, Security**

I. Introduction

One drawback of Public Key Infrastructure (PKI) is that all devices in the network use public/private key pairs, and the private key is typically stored in non-volatile memory (e.g., disk). If a private key is recovered from a device by an attacker, then they are able to masquerade as the user of the private key.

Several efforts have been proposed to mitigate this drawback by equipping client devices with Physically Unclonable Functions (PUFs) for the purposes of authentication [1], [2]. PUFs are volatile memory and are unique due to variance in the manufacturing process, allowing each PUF to act as a unique fingerprint for a device. PUFs are employed to generate random numbers (hereafter referred to as seeds) that are used as input into cryptography algorithms. The cells in a PUF are unstable and drift over time due to environmental factors such as temperature [3]. This error needs to be corrected, otherwise authentication will fail. One method that has been proposed to mitigate this error is to use error correction codes (ECC), data helpers, and fuzzy extractors [4]–[7] but they have two major drawbacks: (i) many devices are low-powered and are unable to correct for the error due to latency and/or power constraints; and, (ii) these mechanisms leak information [8]–[10] about the PUF if the device is compromised.

To address the drawbacks of error correction codes, the Response-Based Cryptography (RBC) protocol was developed [1], [11]–[15] which places the computational burden of authenticating PUF-equipped client devices on a secure server such that low-powered client devices do not need to employ error correction codes, data helpers, or fuzzy extractors. In the RBC algorithm, during enrollment, a server records the PUF images that are deployed in client devices. During the handshake between the server and client, the server instructs the client to read a subset of its PUF cells (this is denoted a PUF challenge), and the client uses this to generate a seed (a PUF response) which is used to create public/private key pairs. The client sends the public key to the server for authentication and the server generates a public key from the client PUF image. Because the seed generated by the PUF may have drifted and has a bit error rate (BER) $>$0% relative to when the PUF was enrolled, the server performs a search by flipping bits in the seed generated from the PUF image such that it reproduces the client's public key. If it does so within a Hamming distance threshold, the client is authenticated.

Lee et al. [16] proposed an efficient search method for iterating over PUF seed spaces. The search method employs the probability that a given cell in the PUF is likely to flip (this is information obtained during the PUF enrollment process), such that an ordering of candidate seeds is exploited. This allows searching the PUF seed space intelligently instead of searching without any knowledge of the probability that a PUF cell will flip. This significantly improves the performance of the RBC search by reducing the total number of seeds that need to be searched and allows for authentication even when PUFs may have drifted significantly from their initial state.

The research on RBC to date, including the most recent by Lee et al. [16] does not permit any error in the transmission be-

Research was sponsored by the Army Research Laboratory and was accomplished under Cooperative Agreement Number W911NF-23-2-0014. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein.

tween the client and server. Securing communication in noisy or low signal environments is critical for many applications. Examples include unmanned autonomous vehicle (UAV) drone package delivery [17] in low signal environments, or wireless communication at a music concert where numerous people are accessing the same network. Securing communication in these settings is challenging for cryptography because by definition, cryptography primitives do not tolerate any error. A *motivating example* is as follows: Consider the following canonical scenario employing *Alice* and *Bob* that wish to communicate. Assume an asymmetric public key infrastructure (PKI) system where Alice wishes to receive an encrypted message from Bob. Alice generates a public/private key pair and sends her public key to Bob but the public key contains an error (i.e., one bit has flipped) due to noise in the transmission environment. Bob encrypts and sends Alice a message using her public key; however, Alice is unable to decrypt the message with her private key due to the erroneous public key received by Bob. This example illustrates that if any of the bits are incorrect, then secure communication is impossible between two parties. While ECC can be used to address this issue it burdens the low-powered device with more work while also enabling attackers to exploit vulnerabilities in ECC information leakage. By correcting for the noisy environment on the server and not the client, we gain an additional layer of security.

To address communication in noisy environments, the contributions of this paper are two-fold. First, we propose an authentication protocol that is resilient to noisy environments. Second, we optimize the system through algorithmic innovations and the parallelization potential of modern multi-core CPUs which are needed to reduce authentication latency. The same benefits of RBC are retained in our protocol, which include: requiring that the secure server perform error correction, thus allowing client devices to be lightweight/low-powered, and employ a probabilistic search of the PUF seed space. Our protocol bolsters the prior work on RBC by extending it to noisy or low-signal communication environments. To summarize, we make the following contributions.

- We introduce Noisy Probabilistic Response-Based Cryptography, NPRBC, a protocol to authenticate low powered devices in high noise environments.
- We evaluate the protocol using a Monte Carlo approach that spans PUF BER noise levels between 20-45 bits and transmission errors (TBER) up to 44% (the upper bound is 50% [18]). The PUF BER is derived from enrollment data of an SRAM PUF and the TBER is selected in intervals that represent varying levels of noise in the environment.

The paper is organized as follows: Section II outlines the proposed protocol, NPRBC, Section III presents the experimental evaluation, and finally, Section IV concludes the paper and discusses future work directions.

II. NPRBC

Our response-based cryptography protocol, NPRBC, builds on the work of Lee et al. [16] which showed that response-based cryptography can employ knowledge of enrollment data which quantifies the bit error rate (BER) of each cell in a client's PUF and stores this information in the PUF image on a secure server. Our proposed protocol differs from Lee et al. [16] as it is robust to noise and requires that the client send a SHA3-512 message digest (M_1) of the hashed 256-bit seed and also a ciphertext (C_1) generated with AES256 using the same 256-bit seed as the key. Sending both M_1 and C_1 allows the protocol to better identify the candidate seed which the client used to generate M_1 and C_1 rather than using M_1 alone, as was the case in Lee et al. [16].

We outline the steps of our protocol, NPRBC, which are illustrated in Figures 1 and 2. The steps in the protocol pertaining to Figure 1 are as follows:

1) Client/Server: The server/certificate authority (CA) performs a handshake with the client and tells it which cells to challenge in its PUF.
2) Client: The client uses this information to generate a 256-bit seed (S_1) for two purposes in Steps 3–4 below.
3) Client: S_1 is salted and then is used to generate a public/private key pair (P_{k1}/P_{r1}).
4) Client: S_1 is used as input to create a message digest using SHA3 (M_1) and ciphertext generated by AES256 (C_1).
5) Client: M_1 and C_1 are sent to the server for authentication.
6) Server: The server performs the RBC search (outlined in the steps below). After the search finds the most likely seed, S', generated by the client it is used as input to a cryptographic algorithm (e.g., ECC, or a post-quantum cryptography algorithm), the client's public key is generated (P_{k1}) and registered on the registration authority.

The steps in the protocol pertaining to Figure 2 (the RBC Search Engine in Figure 1) are as follows:

1) The server receives M_1' and C_1' from the client.
2) The server reads the initial seed from the client's PUF image (S_{init}).
3) The server hashes S_{init} to create a message digest M and checks if it matches M_1' ($M \approx M_1'$) received from the client. Recall that M_1' is the corrupted variant of M_1 (see Section II-C for details).
4) If $M \not\approx M_1'$, then go to Step 10.
5) If the algorithm continues, then the next seed is generated and the process restarts at Step 3 above, but S is permuted and then hashed (S_{init} is only used on the first iteration).
6) If $M \approx M_1'$ then we generate the ciphertext, C.
7) The ciphertext, C, is checked for a match with the ciphertext received by the client ($C \approx C_1'$). Recall that C_1' is the corrupted variant of C_1 (see Section II-C for details).
8) If $C \approx C_1'$ then the seed, S_c, is added to the candidate set and the termination criteria are checked and if the algorithm continues then go to Step 5 above, otherwise go to Step 10.
9) If $C \not\approx C_1'$ then go to Step 10.
10) The termination criteria are checked (see Section II-D for details). If the algorithm terminates, then all the candidate seeds are checked and those with the highest probability of being correct are selected (see Section II-E for details). A seed is selected (S_s) and is salted to create S'.

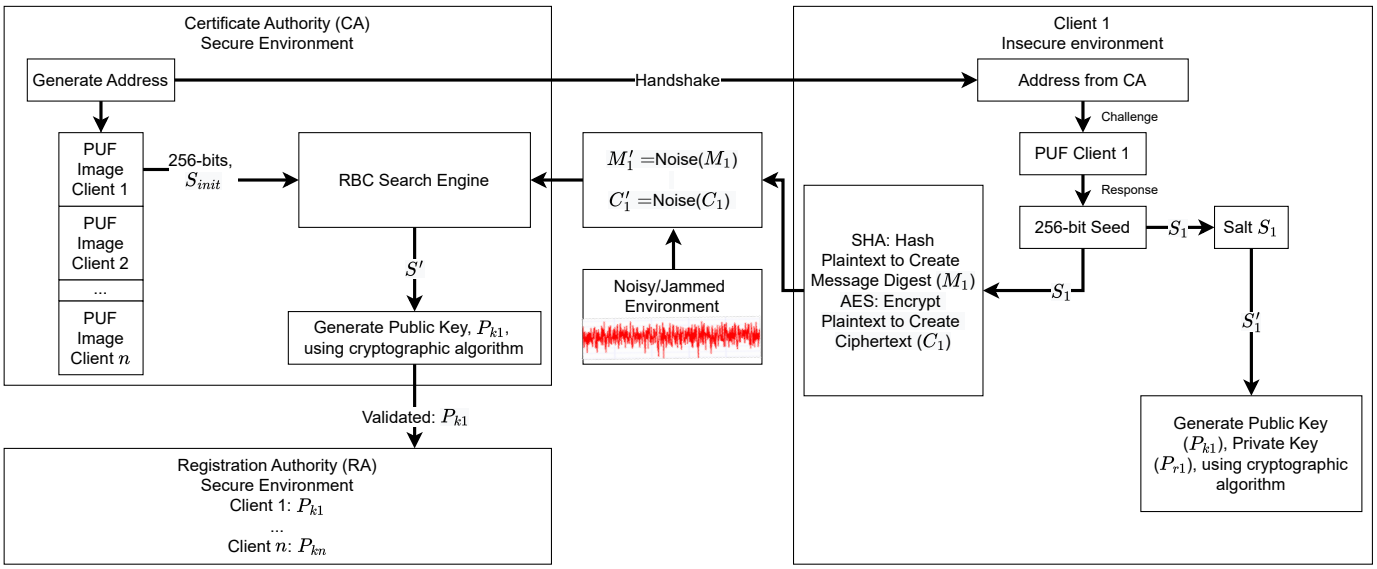

Fig. 1. The response-based cryptography protocol that is robust to noise (NPRBC). The RBC search engine is shown in Figure 2. This inspiration for this figure is from similar figures in the literature [16], [19].

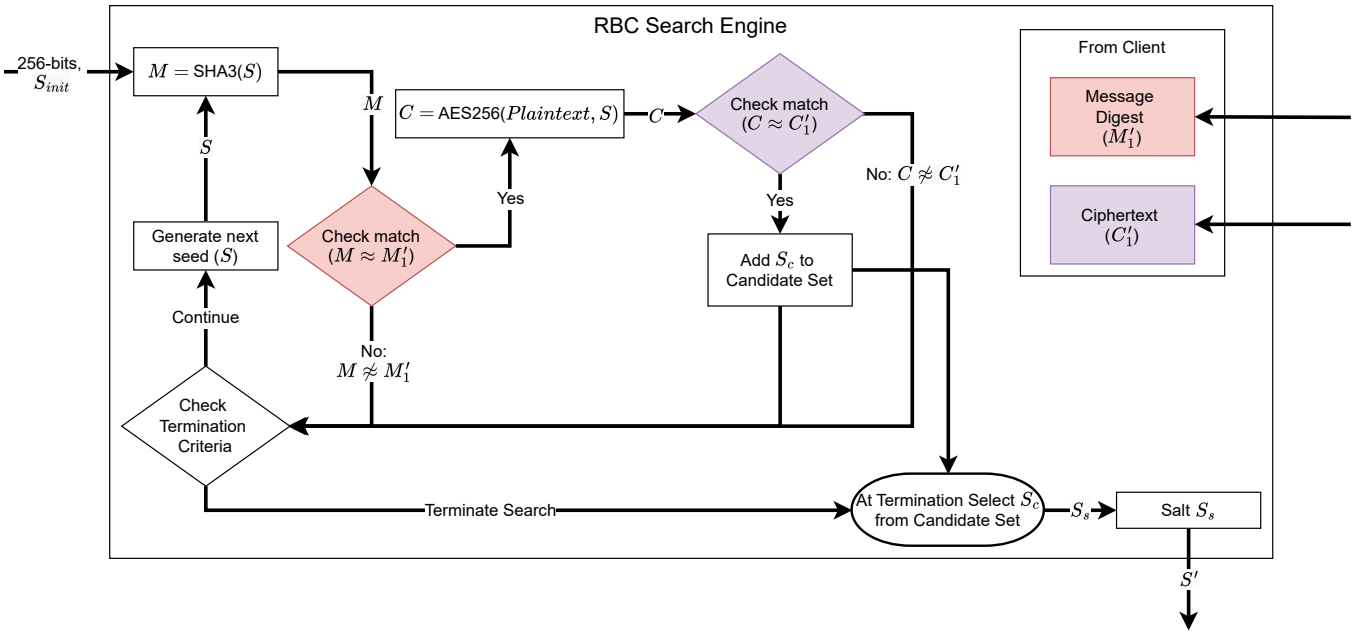

Fig. 2. The probabilistic RBC search engine. Colors represent comparisons between client and server information.

A. Probabilistic Searches of the PUF Seed Space

The cells of the PUF generate a response when challenged, where each cell's response is either a 0 or 1. During enrollment, the cells of the PUF are challenged numerous times and the responses are recorded to produce a PUF image. In the PUF used in our evaluation, most cells are stable; Figure 3 shows that the PUF has 573 bits that do not change between consecutive challenges, but some cells have a porbability of yielding either 0 or 1. During the authentication process, both the server and the client select the same cells (and in the same order) for their respective seeds based on the handshake that is initiated by the server. If all of the cells selected are stable, then the client and the server share the same initial seed. If cells are selected that vary between challenges then there is a probability that the server's and client's seeds will not match. In such a case, a search is performed on the server to determine which bits have drifted in the client's PUF relative to the image stored on the server. The drift of a seed is the number of bits that differ between the client's seed and the seed image on the server (this is the Hamming distance between the bits of the client's seed and the bits of the seed image).

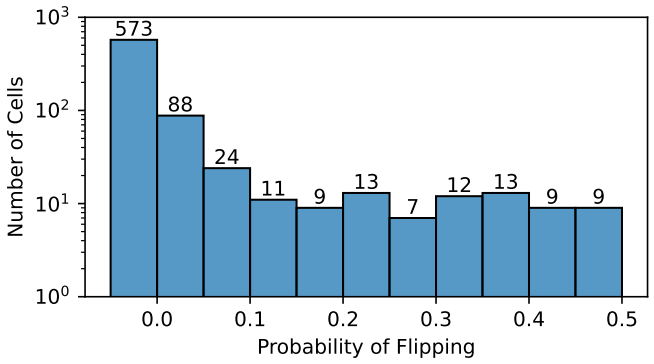

Fig. 3. This histogram shows the probability of a bit flipping for the PUF used in our experimental evaluation, which is an ISSI 61-64WV6416BLL SRAM chip. There are a total of 768 enrolled cells that are stored in the PUF image on the secure server. While 573 of those cells are stable and will not change between authentication sessions, the other 185 cells have up to a 0.5 (or 50%) probability of their state changing when challenged during authentication.

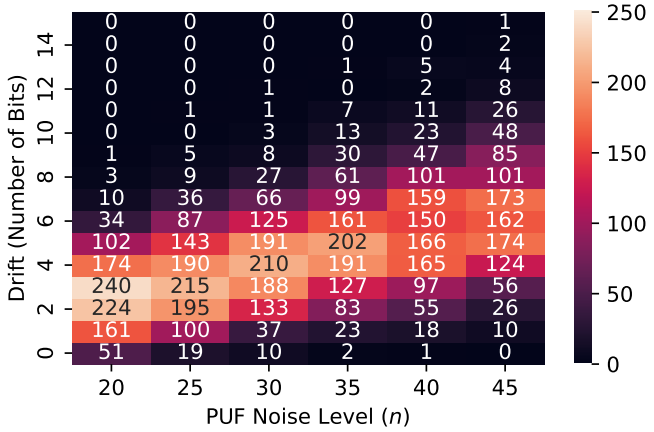

Fig. 4. This heatmap plots the number of samples at each drift value for each PUF noise (n) using the PUF outlined in Figure 3. There are 1,000 trials for each n value with each trial selecting unstable cells at random from the PUF. The intensity of the heat map is the number of trials for each selected n value that had the corresponding amount of drift.

PUF Noise Level (n) – As described above, the server selects which addresses the client reads from its PUF. Therefore, the server selects some number of unstable cells (cells having a non-zero probability of flipping). We denote the number of unstable cells selected to be n (which is the PUF noise level). For example a PUF noise level of $n = 20$ indicates that a seed is generated using 20 cells from the PUF that are not stable and have a chance of producing a different state compared to the image on the server. Figure 4 shows the amount of drift that occurs with varying PUF noise level values (n) for the PUF used in our evaluation. For example, for $n = 20$, there were 240 seeds out of 1000 where there was 3 bits of drift from the server's seed recorded in the PUF image (this is a Hamming distance of 3 between the PUF image and client's seed). As n increases, the amount of drift increases as well. Individual seeds with higher drift are less likely to occur, but at each level of drift there are exponentially more possible seeds, so while the individual seed with that level of drift has a low probability of occurring, the set of seeds with that level of drift may be more likely. This explains why higher noise levels ($n > 30$) rarely have seeds with low (≤ 1) drift levels.

Accumulating Probability – The probability of each bit flipping in the client's seed is determined from the enrollment data on the server. This information is used to determine the likelihood of an individual seed being generated from the set of selected cells. The single most likely seed to be correct is the one where no bits have drifted relative to the server's image. As the probabilistic search proceeds, the probability of each searched seed is added to a sum. This sum yields the total probability of the correct seed having been found. Individual seeds that have more drift, i.e. the number of bits in the client's seed that have flipped, are less likely to occur than seeds with low drift.

Search Order – The probabilistic search checks seeds in order of highest to lowest probability of matching. This allows the search to prioritize the seeds which have a higher chance of being correct while disregarding seeds that have an infinitesimally small probability of being correct (e.g., a seed with 15 bits that drifted would never be searched because the probability of that occurring is too low to be worth considering). This search order leads to probability accumulating quickly at the beginning of the search and then to diminishing returns as the search continues. This search strategy is much more work efficient than prior work that assumes all bits in a seed have the same probability of flipping [1], [11]–[15].

B. Noisy Transmission

The transmission from the client to the server is vulnerable to corruption by environmental noise. Our protocol allows for the client to still be authenticated despite high environmental noise which flips bits in both the message digest and ciphertext. The protocol accepts a transmission bit error rate (TBER) up to a set threshold, t, in both the message digest and the ciphertext to account for corruption during transmission.

C. Match Criteria

A noisy/jammed environment will corrupt the signal from the client to the server resulting in M_1' and C_1' which are corrupted variants of the intended transmissions. The Hamming distance, d, between the corrupted variants and server generated variants is used to determine their similarity such that $d_M = \text{dist}(M, M_1')$ and $d_C = \text{dist}(C, C_1')$ where the dist() function is used to compute the Hamming distance between two sets of bits. The match criteria sets the threshold value, t, so that if $d \leq t$, then the match is considered true and the seed, S which is was used to generate M and C is added to the candidate set. This allows the RBC Search Engine to account for a given amount of noise during transmission and to still authenticate the client without having a perfect match, where $d_M = d_C = 0$. The server continues to generate seeds and match them until a termination criterion has been met.

TABLE I
PARAMETER VALUES USED IN THE EVALUATION. VARIED REFERS TO WHETHER THE PARAMETER IS VARIED IN THE EVALUATION.

Parameter	Value	Varied
Probability Threshold (t)	0.999	
Time Limit	5 s	
TBER	10-44%	✓
PUF Noise Level (n)	20-45 bits (7.81-17.6% BER)	✓

D. Termination Criteria

There are two factors that determine when the search terminates. First, a probability threshold of 0.999 is used to terminate the search once the sum of the probabilities of the searched seeds reaches the threshold. Second, a time limit is used to terminate the search in cases where the sum of the probabilities has not accumulated to the threshold of 0.999. After the termination criteria has been met, the server then selects the best seed from the candidate set.

E. Candidate Seed Refinement

When the server accepts a high level of noise during transmission a large number of seeds are added to the candidate set. To choose the correct seed, S, from all of the candidates the server selects the seed with the lowest overall transmission error, $d_{total} = d_M + d_C$. The probabilistic search only examines a small fraction of the total possible seeds and so the probability of an incorrect seed on the server creating both a message digest (M) and ciphertext (C) that is corrupted during transmission to be closer to the client's M_1' and C_1' than the correct seed's M and C is negligible. Requiring C_1' and lengthening M_1' and C_1' increases the probability of identifying the correct seed in high TBER scenarios.

III. EXPERIMENTAL EVALUATION

A. Experimental Methodology

All experiments are conducted on a platform containing 2× AMD EPYC 7542 CPUs (64 total physical cores) clocked at 2.9 GHz with 512 GiB of main memory. All code is written in C/C++. All executions of the program are parallelized using OpenMP and employ 64 threads/cores as we found this to achieve the best performance on our platform.

PUF Used in the Evaluation – We use the SRAM PUF with characteristics outlined in Figures 3 and 4. In the protocol we select up to $n = 45$ unstable bits out of a 256 bit seed, where the remaining bits are stable. Note that while there are numerous PUF technologies, the SRAM PUF employed here has very high levels of noise. For instance, a Magnetic RAM PUF yielded a 7.7% BER [20], which is lower than the 17.6% BER here (or $n = 45$ of 256 bits). Also, a Resistive RAM (ReRAM) PUF has been reported to have a BER of 0.001 (only 1 in 1000 bits are unstable) [2]. Thus, our PUF is representative of having a substantial BER which is the worst case scenario for the protocol. Consequently the results are applicable to PUFs with similar or lower levels of noise.

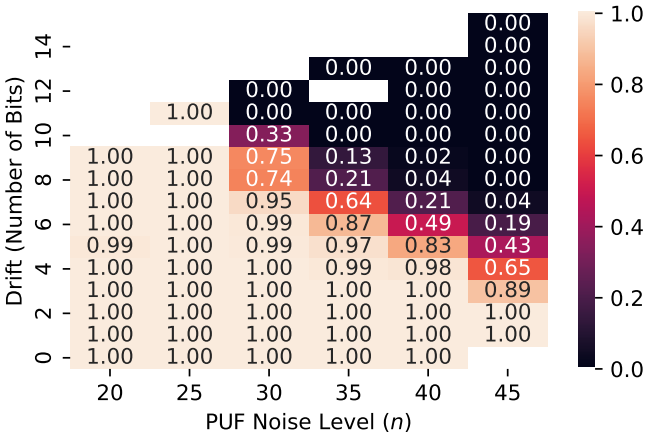

Fig. 5. The heatmap plots the success rate of authentication for each PUF bit error rate (n) and drift with a TBER of 30%. Blank values in the heatmap indicate that there were no trials that had that combination of n and drift (see Figure 4). A success rate of 1.00 indicates that it always succeeded to authenticate with the given parameters, while a success rate of 0.00 indicates that it never succeeded to authenticate.

Fixed Parameters for Experimentation – A number of parameters are fixed for experimentation to allow for a detailed examination of parameters that significantly change the authentication rate of the protocol. The threshold, t, of transmission noise acceptance is fixed to 50% to account for TBER $\leq 44\%$ (see Section II-B). Lower threshold values do not improve authentication rates (nor do they degrade it as long as they are $\geq 5\%$ above the TBER) while higher threshold values ($t \geq 50\%$) result in a lower authentication rate. A fixed time limit of 5 seconds and an accumulated probability of 0.999 is used as the termination criteria (see Section II-D). A ciphertext of 1024-bits is used as it was experimentally found to have the best results on our platform. Additionally, SHA3-512 is used instead of SHA3-256 because the longer message digest increased the authentication rate in our experiments.

Monte Carlo Parameter Sampling – We employ a Monte Carlo approach where each combination of parameters given in Table I is trialed 1000 times. Each trial uses the trial number (1 to 1000) to set the seed for the random number generator. The random number generator determines which cells are selected for the seed and which bits flip during transmission (based on the selected TBER). The number of bits that flip during transmission follows a binomial distribution centered on the average number of bits that flip due to the TBER. This is modeled as Additive White Gaussian Noise (AWGN) resulting in a maximum TBER of 50%, which is Shannon's Limit [18].

B. Experimental Results

We simulate different PUF seeds with the Monte Carlo approach described above and report the results in Table II. The results show that, up to a TBER of 30%, the determining factor for a successful authentication is the amount of PUF noise (n). Noise levels $n = 20 - 30$ bits with TBER $\leq 30\%$ successfully authenticate $\geq 98\%$ of the time. As the n value

TABLE II
AVERAGE EXECUTION TIMES AND AUTHENTICATION RATES FOR PUF NOISE LEVELS $n = 20 - 45$, TBER LEVELS 10 – 40% AND TIME LIMIT OF 5S.

PUF Noise (n)	20 Bits				25 Bits				30 Bits				35 Bits				40 Bits				45 Bits			
TBER (%)	10	20	30	40	10	20	30	40	10	20	30	40	10	20	30	40	10	20	30	40	10	20	30	40
Time (s)	0.37	0.36	0.37	0.37	1.60	1.60	1.62	1.61	4.56	4.56	4.56	4.56	4.56	4.56	5.00	5.00	5.00	5.00	5.00	5.00	5.00	5.00	5.00	5.00
Success	0.99	0.99	0.99	0.88	0.99	0.99	0.99	0.77	0.98	0.98	0.98	0.67	0.84	0.84	0.84	0.57	0.58	0.58	0.58	0.39	0.28	0.28	0.28	0.00

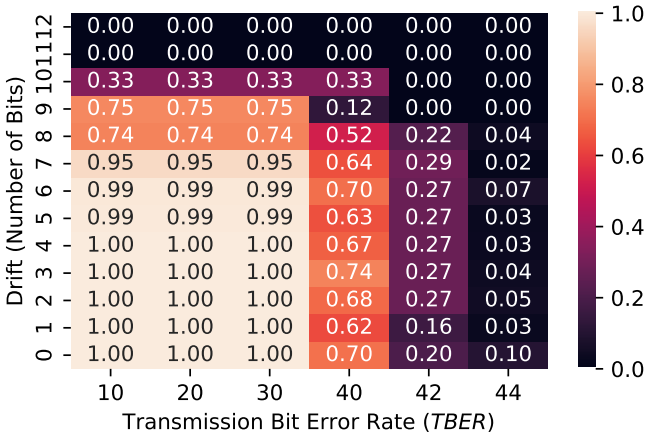

Fig. 6. The fraction of successful authentications are plotted vs. the TBER (the chance of each transmitted bit flipping in the message digest or ciphertext) for a PUF noise level (n) of 30%.

increases to 35, 40 and 45 bits the fraction of successful authentications with TBER of 30% drops from 0.98 to 0.84, 0.58, and 0.28, respectively. Additionally, the average time to complete the search increases as a function of n. The time limit only impacts the success rate of the searches when $n \geq 35$. Recall from Figure 4 that the higher the PUF BER (n), the higher the average drift. The increase in drift requires more computation to find the correct seed. This is reflected in the average execution time values in Table II. In the scenarios where the search is terminated by the time limit, the probability of finding the correct seed is lower because an insufficient number of seeds are searched.

In Figure 5 the success rate is plotted for each n and corresponding drift value. Every seed with a drift of 2 or lower is found while every seed with a drift of 12 or higher is not found. Additionally, the higher n values fail to find the correct seed at drift levels where lower n values succeed. This is due to the probabilistic search having to search through more possibilities at each drift value. From Table II we observe that this results in the time limit of 5 seconds being reached, and the search terminating before the correct seed is identified.

While a TBER below 30% does not impact the authentication rate for a given n value, as the TBER increases past 30% it begins to be the limiting factor in authentication as opposed to n. Recall from Section II-E that the seed which is selected from the candidate set is the one with the lowest transmission error. In Figure 6 we observe that as the TBER approaches 44% the success rate rapidly decreases. We found that this is because we cannot identify and select the correct seed from the candidate set because of the high transmission error. High TBER compounds the issue with high n, resulting in poor authentication rates for seeds generated with a high n coupled with a high noise environment. The protocol mitigates this by selecting PUF cells that generate seeds with lower noise levels (n) in environments with high transmission bit error rates.

Leveraging Compute Power for Security – The two factors contributing to the success rate of NPRBC are the PUF Noise and the TBER. Increasing the compute power available increases the number of seeds searched which increases authentication rates at higher n values. This has the added affect of increasing the overall security because a higher n value (corresponding to a noisier PUF) is harder to attack. Increasing the time limit also allows more seeds to be searched but gives attackers additional opportunity to compromise the system and adds latency to time sensitive communications. The more compute power available on the server, the more secure NPRBC becomes. We reiterate that because the server has secret information regarding the client (the PUF image), compared to the number of seeds searched by the server, the search space for an attacker is intractable (2^{256} seeds).

IV. DISCUSSION & CONCLUSION

NPRBC enables rapid device authentication in congested electromagnetic environments that have been previously found to be too noisy for canonical (zero-noise tolerance) cryptographic protocols [1], [11]–[15], [19]. We evaluated NPRBC using a range of PUF noise levels $n = 20-45$ and transmission bit error rates (TBER) of 10–44%. Recall that the 44% limit refers to where each bit in the transmission of the message digest or ciphertext has a 44% chance of flipping. We evaluated the protocol by using a Monte Carlo approach that varied which bits are selected from the PUF and which bits flip in the transmitted data (message digest and cipertext). With low PUF noise ($n \leq 30$), the protocol is successful in almost every trial. Non-probabilistic search methods are unable to authenticate in as noisy an environment as NPRBC because they search too many seeds which increases the difficulty of distinguishing the correct seed from all of the candidate seeds. Additionally, non-probabilistic searches are intractable for seeds with higher drift values as shown in previous work [19]. NPRBC successfully authenticates with a drift of up to 11-bits which for previous works requires more than 6.2×10^{18} seeds to be searched and would require >20 years to authenticate on a modern multi-GPU server node.

REFERENCES

[1] B. Cambou, "Unequally Powered Cryptography With Physical Unclonable Functions for Networks of Internet of Things Terminals," in *2019 Spring Simulation Conf.*, 2019, pp. 1–13.

[2] B. Cambou and S. Jain, "Key Recovery for Content Protection Using Ternary PUFs Designed with Pre-Formed ReRAM," *Applied Sciences*, vol. 12, no. 4, p. 1785, 2022.

[3] S. Taneja, A. B. Alvarez, and M. Alioto, "Fully synthesizable PUF featuring hysteresis and temperature compensation for 3.2% native BER and 1.02 fJ/b in 40 nm," *IEEE Journal of Solid-State Circuits*, vol. 53, no. 10, pp. 2828–2839, 2018.

[4] M. Hofer and C. Böhm, "Error Correction Coding for Physical Unclonable Functions," in *Austrochip, Workshop on Microelectronics*, 2010.

[5] J. Delvaux, D. Gu, D. Schellekens, and I. Verbauwhede, "Helper Data Algorithms for PUF-Based Key Generation: Overview and Analysis," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, no. 6, pp. 889–902, 2014.

[6] G. T. Becker, A. Wild, and T. Güneysu, "Security Analysis of Index-Based Syndrome Coding for PUF-Based Key Generation," in *2015 IEEE Intl. Symp. on Hardware Oriented Security and Trust*, 2015, pp. 20–25.

[7] B. Fuller, X. Meng, and L. Reyzin, "Computational fuzzy extractors," *Information and Computation*, vol. 275, p. 104602, 2020.

[8] J. Darbon, B. Sankur, and H. Maitre, "Error correcting code performance for watermark protection," in *Security and Watermarking of Multimedia Contents III*, vol. 4314. SPIE, 2001, pp. 663–672.

[9] S. A. Vanstone and P. C. Van Oorschot, *An introduction to error correcting codes with applications*. Springer Science & Business Media, 2013, vol. 71.

[10] 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 *14th Intl. Wireless Communications & Mobile Computing Conf.* IEEE, 2018, pp. 1261–1266.

[11] C. Philabaum, C. Coffey, B. Cambou, and M. Gowanlock, "A Response-Based Cryptography Engine in Distributed-Memory," in *Intelligent Computing*, K. Arai, Ed. Cham: Springer Intl. Publishing, 2021, pp. 904–922.

[12] J. Wright, Z. Fink, M. Gowanlock, C. Philabaum, B. Donnelly, and B. Cambou, "A Symmetric Cipher Response-Based Cryptography Engine Accelerated Using GPGPU," in *2021 IEEE Conf. on Communications and Network Security*, 2021, pp. 146–154.

[13] B. Cambou, M. Gowanlock, B. Yildiz, D. Ghanaimiandoab, K. Lee, S. Nelson, C. Philabaum, A. Stenberg, and J. Wright, "Post Quantum Cryptographic Keys Generated with Physical Unclonable Functions," *Applied Sciences*, vol. 11, no. 6, 2021.

[14] K. Lee, M. Gowanlock, and B. Cambou, "SABER-GPU: A Response-Based Cryptography Algorithm for SABER on the GPU," in *2021 IEEE 26th Pacific Rim Intl. Symp. on Dependable Computing*, 2021, pp. 123–132.

[15] J. Wright, M. Gowanlock, C. Philabaum, and B. Cambou, "A CRYSTALS-Dilithium Response-Based Cryptography Engine Using GPGPU," in *Proc. of the Future Technologies Conf. 2021, Volume 3*, K. Arai, Ed. Cham: Springer Intl. Publishing, 2022, pp. 32–45.

[16] K. Lee, B. Donnelly, M. Gowanlock, and B. Cambou, "Efficient searches of physical unclonable function seed spaces for response-based cryptography," in *Autonomous Systems: Sensors, Processing and Security for Ground, Air, Sea, and Space Vehicles and Infrastructure 2023*, vol. 12540. SPIE, 2023, pp. 112–125.

[17] S. A. H. Mohsan, M. A. Khan, F. Noor, I. Ullah, and M. H. Alsharif, "Towards the Unmanned Aerial Vehicles (UAVs): A Comprehensive Review," *Drones*, vol. 6, no. 6, 2022.

[18] C. E. Shannon, "Communication in the presence of noise," *Proceedings of the IRE*, vol. 37, no. 1, pp. 10–21, 1949.

[19] K. Lee, B. Donnelly, T. Sery, D. Ilan, B. Cambou, and M. Gowanlock, "Evaluating accelerators for a high-throughput hash-based security protocol," in *Proc. of the 52nd Intl. Conf. on Parallel Processing Workshops*, 2023, pp. 40–49.

[20] A. Nejat, F. Ouattara, M. Mohammadinodoushan, B. Cambou, K. Mackay, and L. Torres, "Practical experiments to evaluate quality metrics of mram-based physical unclonable functions," *IEEE Access*, vol. 8, pp. 176 042–176 049, 2020.