Physically Unclonable Functions with multi-states and Machine Learning

Bertrand Cambou, and Fatemeh Afghah
Northern Arizona University, Flagstaff, Arizona, USA
Bertrandc.cambou@nau.edu; fatemeh.afghah@nau.edu

Abstract: When subject to natural effects such as aging, temperature changes, bias voltages drifts, or electrostatic interferences, the profile of the Physically Unclonable Functions (PUF) Challenge-Response-Pairs (CRPs) error rates is made predictable when analyzed by multi-states and a Machine Learning Engine (MLE). With error correction, physical drifts do not result in false negative authentications (FNA), while statistically abnormal CRPs are flagged without increasing the risk of false positive authentications (FPA). PUFs that are hard to uncover by side channel analysis that would be normally weak become excellent candidates.

Keywords: cryptographic primitives; Physically Unclonable Functions; hardware authentication; multi-state architecture; coding-decoding methods; machine learning.

1. BACKGROUND

Physically Unclonable Functions (PUFs) are strengthening authentication methods, and this as part of a set of cryptographic primitives. PUFs exploit intrinsic manufacturing variations naturally, which are introduced during the fabrication of the devices such as critical dimensions, doping level of semiconducting layers, and threshold voltages [1 to 5]. These variations make each device unique, and identifiable from each other. The underlying mechanism of PUF is the creation of a large number of Challenge (i.e. Input) Response (i.e. output) Pairs (called CRPs) that are unique to each device. Once deployed during the authentication cycles, the PUFs are queried with challenges. The authentication is granted when the rate of matching responses is statistically high enough.

Virtual Machine and Machine Learning:

Virtual Machines, and Machine Learning Engines (MLE), are used in cyber-security [6 to 10] on the terminal side of a Cyber Physical System to prevent attacks. These MLEs need to operate in close loop, without external intervention to avoid disclosing additional information during attacks. The MLEs can be dedicated to handle security, and authentication isolating the terminal in case of a successful download of a malware. Current encrypted secure elements have both the computing power, and the secure memory space to support sophisticated Virtual Machine, and MLE functions.

Error Correction:

Different error correction coding (ECC) techniques have been utilized to reduce the intra-PUF variation factor in order to improve the similarity of the PUF responses to the same challenge in different attempts. The existence of deterministic noise suggests the inefficiency of repetition coding (ARQ), hence forward error codes (FEC) are being used in PUF systems to improve their performance. The idea of using ECC is borrowed from communication systems, where redundant information (parity or helper data) is added to the input signal to provide the possibility of error detection and correction over a noisy channel. Linear block codes have been widely utilized for error correction in several PUF types (Price and Sherman n.d.[11]), (Boehm 2010[12]) (Yu and Devadas 2010[13]) (Herder, Yu and Koushanfar 2014[14]). One of earliest attempts of using ECC was implementation of 2-D Hamming codes with low error correction capability (B. Gassend 2003[15]). Reed-Solomon code is a class of linear block codes with a good performance in combating burst noise, hence considering a fairly uniform distribution of noise in PUF applications will not be an acceptable candidate; where Binary Bose-Chaudhuri-Hochquenghen (BCH) codes with fuzzy extractor have been used in (J. Guajardo, et al. 2007[16][17]). A (255, 63, t=30) BCH code in which 192 syndrome bits out of the n=255 codeword length were exposed publicly was introduced in (Suh 2005[18]). This code offers a PUF with about 88% stability that has the capability of correcting at most 30 errors out of 255. In (Maes, Tuyls and Verbauwhede 2009[19]), the authors proposed a soft decision making method for helper data algorithms in SRAM-based PUFs, where a linear block coding as concatenation of repetition, BCH and Reed-Muller codes with soft decision maximum-likelihood decoding was utilized. It was shown in (Maes, Tuyls and Verbauwhede 2009[19]) that the soft helper data will not reduce the min-entropy. It is worth noting that some high performance coding techniques such as convolutional coding and Low Density Parity Check (LDPC) coding will be appropriate for this application as they need a very long data string in order to achieve their efficient performance, hence will be applicable to PUF error correction. However, these commonly used linear error correction codes cannot overcome the high data error rate of up to 25% in subsequent read-outs of PUF in the presence of extreme external variations. These methods are using a hard-decision decoder to estimate the new readouts data in related to the initial data string that can result in a considerable information lost.

2. GENERAL DESCRIPTION

The method described in this paper is to design a PUF with multi-state and Machine Learning Engine (MLE) based on (i) a Challenge or Response generation process that captures the specific “personality” of the physical elements underling the PUFs in a multi-state memory; (ii) an authentication process that quantifies the profile of the CRP error rates, as well as the surrounding input parameters (such as ambient temperature); and then (iii) a computation by the MLE to finalize the authentication process, the coding-decoding algorithms are described in section 2 . Examples described below sometime assume that the size of the PUF challenges are N=128 bits, and that the Challenges and Responses are generated by a memory array based on a particular physical parameter as described in many references [1 to 5]. The method will also be described in much broader terms.

2.1 PUF Challenge /Response generation:

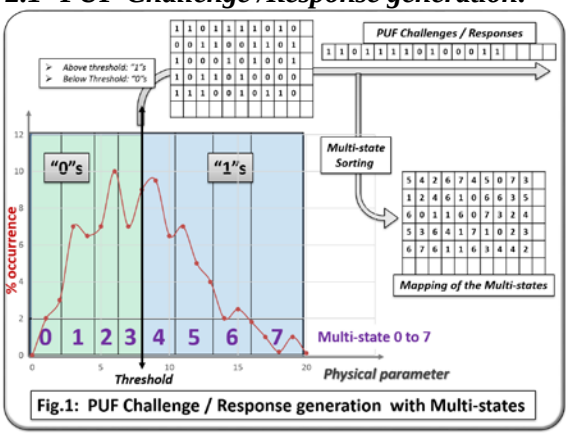


Fig.1: PUF Challenge / Response generation with Multi-states

The Challenge / Response generation process assumed in the example shown in Fig.1 is based on a memory array, and the measurement of a physical parameter. A “0” is programmed in the cells where the parameters are measured below the

threshold located in the middle of the distribution, and a "1" for the cells measured above the threshold. The PUF challenge, a stream of binary bits, is directly extracted from the memory after programming, and sent to secure server. As shown Fig.1, the cells are organized in n multiple states by sorting out the value of the physical parameters underlining each cell. For example the 16 cells with the lowest value are given the state 0, the following 16 cells the state 1, all the way to the 16 cells with the highest value that are the state 7. That way the 128 bits of the PUF are sorted in 8 different states. The precise mapping of the PUF memory array can be stored in a secure memory during the Challenge generation process, this is **Option A**. **Option B**, is to extract the mapping during the Response generation process. In more general terms a PUF of N bits is to be sorted into n states, either during Challenge generation, or Response generation. Each state i is having ni cells in such a way that $\sum_{ni=1}^{ni=n} ni$=N. The block diagram Fig.2 is showing the Challenge generation process.

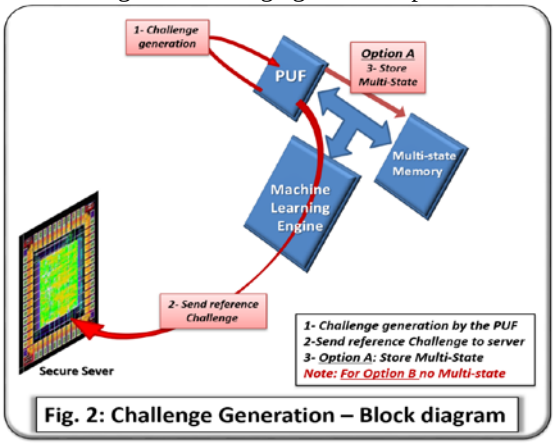


Fig. 2: Challenge Generation – Block diagram

2.2 *PUF Response generations and CRP error rates*

The PUF Responses are generated the exact same way as the Challenges were generated by the PUF memory, and this as often as there is a need for a fresh authentication, however the Responses can vary over time. As shown Fig.3 CRPs errors are to be expected considering that the measurement of the physical parameters underlining the PUFs are naturally evolving over time, such as when subject to external effects such as temperature changes. For a given cell k that is part of the PUF, the CRP error between the Challenge Ck and the Response Rk is given by the equation (1). **Δ**CRPk is the CRP error rate of the cell k. For the ni cells that are part of the state i the average CRP error rate $\boldsymbol{E_i}$ is given by (2). For each response, the Vector of Error VE t is given by (3):

(1) $\Delta CRPk = |Rk - Ck|$

(2) $Ei = \frac{1}{n_i}\sum_{k=1}^{k=n_i} |Rk - Ck|$

(3) $VE = E_0, E_1, .., E_i, .., E_n$

The CRP error rates, and Vector of error computations are valid for both Options. In one case, Option A, the segmentation by a state is done during the Challenge generation process, or during the Response generation process for Option B. VEs are used for the authentication process.

2.3 *MLE for secure authentication*

An MLE is introduced to perform secure authentication as summarized on Fig.4. When the server sends a Challenge to the MLE, a fresh Response is generated by the PUF. The MLE gathers the Response, as well as all available data to compute the authentication. In Option A, the MLE retrieves the mapping of the multi-states from the secure memory. In option B a fresh mapping of the multi-states is generated during the Response generation process. The MLE with a crypto-processor can handle the communication between the secure server and the PUF. The data available for authentication j, as shown in Fig.5, is:

- The Vector of Error: $VE_j = (E_0, E_1, .., E_i, .., E_n)_j$, and this as described in section ii)
- The Vector of Input: $I_j = (I_0, I_1, .., I_l, .., I_m)_j$; this includes all parameters that could be available to the MLE such as operating temperature, biasing voltage and current conditions, and EMI noise.
- The "Learning" data base that incorporates a record of prior Responses,
- The generic predictive models describing the laws of physics underlining the parameters of the PUF. For example the impact of temperature on the parameter is well described by predictive models.

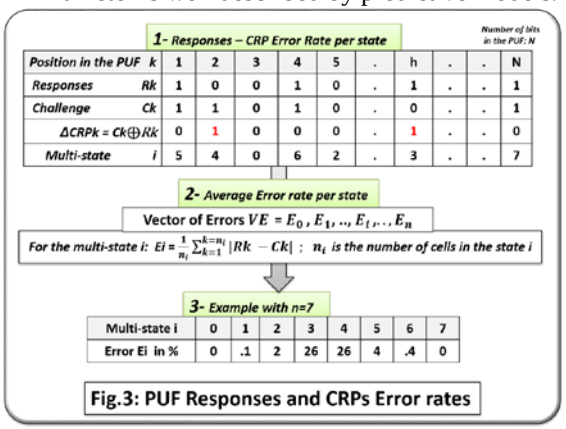


Fig.3: PUF Responses and CRPs Error rates

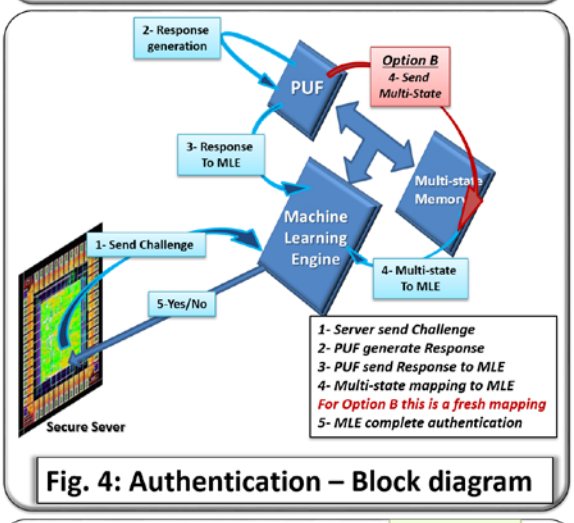


Fig. 4: Authentication – Block diagram

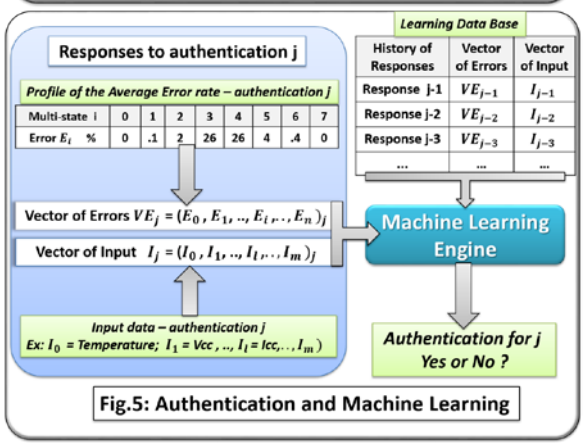


Fig.5: Authentication and Machine Learning

Examples of computation are described below.

3. CODING-DECODING ALGORITHMS

The main design criteria related to the design of Coding-Decoding PUF algorithms is achieving high inter-device and

low intra-device Hamming distances. The inter-device distance is measured as the average Hamming distance between the responses of two PUF devices to the same challenge that shows the uniqueness of PUF responses. However, the intra-device distance measures the average Hamming distance between the responses from a PUF to the same challenge applied at different times and environmental situations. The changes in environmental conditions, and the aging factor, can result in minor mismatches in circuit components, hence the PUF responses to a challenge can be highly affected by noise (Helfmeier, Boit and Tajik [20], [21]). These variations may occur due to random noise at terminal signals (ex: source, drain and gate), changes in temperature, voltage or aging effects (deterministic noise). The natural causes behind the deterministic portion of noise suggests the possibility of learning this behavior over the course of different experiments and use this as a-priori information in error detection and correction for a new experiment. Using the error correction module can also combat the effect of random noise.

3.1 *Error Correction mode and MLE:*

In this work we are combining error correction code and machine learning mechanism to combat both random and predictable potential noise sources, see Fig 6.

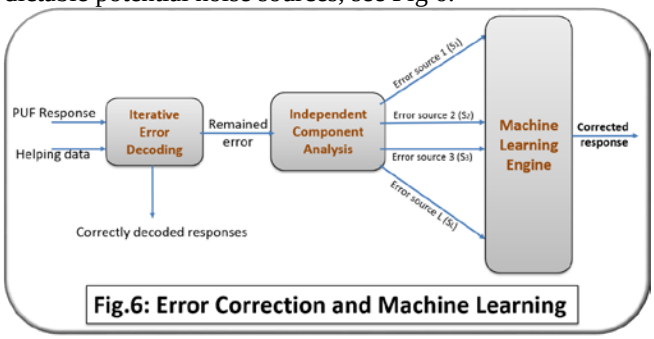


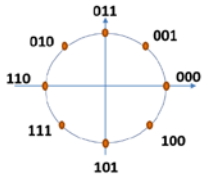
Fig.6: Error Correction and Machine Learning

This mechanism is based on a novel multi-level iterative decoding method to improve the performance of error correction through utilization of intrinsic reliability information in data and also the iterative decoding between the two decoder modules that the soft output of each decoder is fed as a-priori information to other module to improve its decoding accuracy till converging to a desired performance. This design can considerably reduce the error probability. Since a portion of PUF error is due to physical and environmental factors, this can be learned and predicted using the data base information available from experiments on different PUFs. Here we take this knowledge into account to enhance the performance of the proposed method by correcting the remaining error after ECC. Since this error could have been caused due to several factors such as the variations in temperature, voltage and current, we first utilize an Independent Component Analysis (ICA) technique to break down this error to summation of the known possible causes. Independent component analysis is the decomposition of a random vector in linear components that are non-Gaussian and independent or as independent as possible (Hvarinen, and Oja [22]). In the ICA algorithm presented below the assumption of independency among the variables is relaxed to address the possibility of correlation among the physical and environmental factors that may cause variations in PUF readouts. Then each of this error terms are corrected using machine learning algorithm knowing the available data sets associated to variations due to each of these parameters. The PUF readouts are commonly mapped to a binary output, where the number of bits that are different in challenge and response are referred to as CRP error. A measure of PUF error tolerance is determined by Hamming distance. The larger Hamming distance results in lower probability that a noisy readout of a particular PUF will be mapped to another identity in database (lower false acceptance rate). Increasing the length of output bits reduce the false acceptance rate and false rejection rate.

Multi-level soft correction method

The multi-state generation method described in section 1 can be exploited within a novel multi-level soft decision correction method based on iterative decoding, in which the PUF readouts are first quantized to 8 different levels that enables us to have a measurement of readout reliability. Quantizing the readouts to discrete levels helps us to have a measurement of error probability and assignment of Euclidean distance by comparing the readouts to a given reference vector from the challenge. This model is able to further distinguish the accuracy and reliability of the readouts in compared to the binary model since it provides more information regarding the distance of the readouts from the thresholds. In the challenge side, the states are mapped to a binary notation using a Gray coding method, see Fig 7, and next the soft information will be extracted from these. Each bit of the string is coded using two independent BCH coding module based on code-offset technique and the corresponding helping data are encrypted using a Hash function and being available as public information. In the response side, the PUF readouts in the response to the challenge are passed through a similar process of quantization, Gray coding and the extracted information are combined with the two publicly available helping data. The difference between the codewords extracted from the challenge and response is transformed to soft information and is fed to the iterative parallel concatenation decoding structure.

3.2 *Iterative concatenated method*



Decimal	Gray Code	Binary
0	000	000
1	001	001
2	011	010
3	010	011
4	110	100
5	111	101
6	101	110
7	100	111

Fig.7: Gray coding

The novel coding structure is based on multi-state input and an iterative parallel concatenated decoding with soft decision. In hard decision decoding, the output takes a set of possible values (in binary case, 0 and 1), however in soft decoding the output presents the reliability of each bit, e.g. how close the reading is to the thresholds for 0 or 1, instead of saving the results in binary as in hard decision. Therefore, the proposed method can substantially improve the PUF data estimation accuracy by using the intrinsic reliability information in a concatenated structure. Furthermore, the new concatenated method can eliminate the need for having large initial data string to obtain a required accuracy.

- Quantization:

First each readout (Ri) is quantized to a state value from the finite set of (s1, s2, s3,…, sQ), where $Q=2^M$, where Q and M denote the number of non-overlapping states and the number of bits, respectively. The quantization can be performed using common uniform midrise quantization method. Assuming a Gaussian distribution, we utilize the Lloyd-Max quantizer as the optimum design for Gaussian distributed input (Llyod 1982[23]; Yang and and Wu 2012[24]).

- Gray Coding:

In the new method to map the state information to binary data to be transmitted between the server and PUF, we use binary Gray coding in which the adjacent states only differ in one bit. The Gray coding implementation for the case of 8 number of states is depicted in the following figure. This results in enhanced error resiliency of the proposed method.

- Error Control Coding Structure:

A Turbo-inspired coding structure was designed as a parallel concatenation of two coding components Turbo codes have been widely used in communication systems due to their near-Shannon limit performance in noisy and fading channels (Berrou and and Glavieux 1996[25]) (Razi, Afghah and Abedi, Binary Source Estimation Using a Two-Tiered Wireless Sensor Network 2011[26]) (Afghah, Ardebilipour and Razi 2008[27]) (Razi, Ardebilipour and Afghah, Space-Time Block Codes Assisted by Fast Turbo Codes 2008[28]).

The two key features of turbo codes are i) using with a pseudorandom interleaver between the two coding components, and ii) iterative decoding structure with soft-input soft-output (SISO) decoder. The interleaver will distribute the potential error in a dataword over different datawords and consequently increase the probability of error correction. In iterative decoding design, the soft input of one decoder is fed to other decoder and vise verso, till the final decoding result converge with a desired accuracy. The turbo codes are most commonly built up as a parallel concatenation of two Recursive Systematic Convolutional (RSC) codes. However, these will have an efficient performance for a long data string which is not the case in PUF applications. Also for long input length, the computational complexity of the decoding process becomes intractable.

3.2.1 *Coding Structure:*

One approach is to use linear block codes with a turbo-inspired parallel concatenated coding structure and iterative decoding nature to obtain the gain of accurate error correction with an acceptable data string length. Different coding components such as BCH and Reed Solomon (RS) can be utilized in this model. Moreover, the coding can be performed through a multi-stage parallel coding structure. We explain here the case of having two coding components with a random interleaver in between, see Fig. 8A.

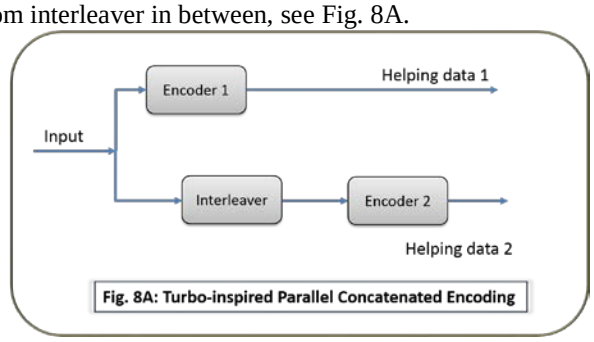


Fig. 8A: Turbo-inspired Parallel Concatenated Encoding

Interleaver: An interleaver is utilized between the two parallel coding components to distribute the possible error over the codeword and produce a randomlike property. This enables us to shuffle the PUF outputs while keeping its linear behavior. Different interleaver designs have been used in communication systems including random interleaver, convolutional interleavers, random interleaver and S-random interleaver. Considering the key-generation concern in PUF application we utilize a random interleaver, where the order for random shuffling is securely saved in PUF to perform the de-interleaving and generate the response and this is not available as public information. The coding structure is designed based on code-offset technique as described in follow. In challenge phase, the PUF readouts are quantized to 8 states and converted to binary format using Gray method to generate bit string $w \in \{0,1\}^n$. Two codewords c_1 and c_2 that are randomly selected from a linear block code set $C_{n,k}$ with minimum distance d (Dodis, Reyzin and and Smith 2004[29]) are added to w. The offset data between w and c_1 and c_2 is called helper data ($h_1=w+ c_1$, $h_1=w+ c_2$)) and is publicly available. In response phase, a fuzzy version $w' \in \{0, 1\}^n$ is generated by PUF, from which c_1' and c_2' are calculated as $c_1'=w'+ h_1$, $c_2'=w'+ h_2$. The distance between c_1' and c_2' with c is used to calculate the reliability information (soft-information). To further enhance the security, the helper that can be encrypted using a Hash function or adding redundant information.

3.2.1 Decoding Structure:

In each round, the extracted soft information is used as an input for a consecutive soft-decision module, hence the PUF key can be determined with a lower length string. The criteria to determine the original codeword associated with current observation from the response is selecting the codeword with minimum Euclidean distance. The reliability of the decoded bits is given by the log likelihood ratio (LLR) of the decision.

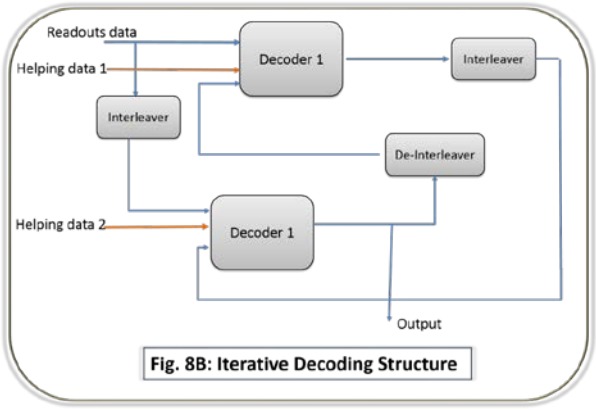


Fig. 8B: Iterative Decoding Structure

A Chase-Pyndiah algorithm is utilized for iterative decoder to minimize the probability of error. In this decoder, each decoder component receives soft input LLR information as logarithm of likelihood ratio (an estimate expressing the probability that the transmitted data bit was equal to zero or one). Both decoder components provide estimates of the same set of data bits in a different order. This information exchange process is continued in an iterative manner till converging with a desired accuracy. At each round, decoders re-evaluate their estimates, using information from the other decoder. The output of the system can be extracted in soft format as the likelihood of each binary bit or as the difference between the original noisy input and the final extracted info, Fig 8B.

4. EXPERIMENTAL VALIDATION

4.1 *Experimental data – Resistive RAM*

In order to model realistic ReRAM PUF CRPs, Cu/TaOx/Pt resistive devices have been fabricated, and characterized at Virginia Tech in a crossbar array on a thermally oxidized Si wafer, Reference [16]. A single Cu/TaOx/Pt switch relies on an electrochemical formation and the rupture of a Conductive Filament (CF) bridging the dielectric between the active Cu and an inert Pt electrode. We are studying the variations of the Vset voltage for the generation of PUF Challenge-Responses-Pairs. Figure 9 (a) shows the cumulative Vset probability distribution within a typical sample of ReRAM memory array, containing 10,000 cells. There exists a minimum Vset voltage applied across the switch, at which a CF is being formed. When the voltage applied to the Cu electrode is pulsed or swept at a positive voltage, the current will remain substantially zero until a critical voltage Vset is

reached, at which a Cu CF is formed connecting the Cu and Pt electrodes, and the cell switches from a high resistive state (HRS) characterized by Roff (1–900 MΩ) to a low resistive state (LRS) characterized by Ron (70–6000 Ω).

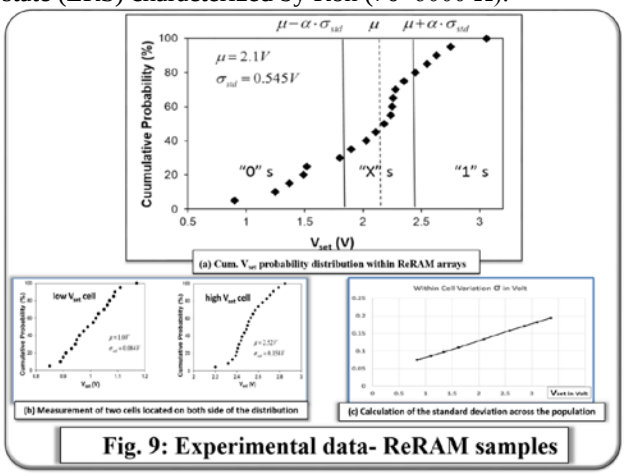


Fig. 9: Experimental data- ReRAM samples

The mean of the Vset distribution is μ=2.1V (indicated by the dashed line) and the standard deviation is **σ**std=0.54 V. In order to study the robustness of the PUF method, and the CRPs error rate, we have characterized the Vset distribution for several individual ReRAM cells. For this characterization, see Figure 9(b), we have selected from the distribution a cell with a low Vset value ($Vset_t \approx 1V$) and a cell with a high Vset value ($Vset_t \approx 2.5V$). The cells have been subjected to repeated reset and set operations under the same conditions. Vset distribution for the low Vset cell is centered around 1V, and its standard variation is **σ**std=0. For the high Vset cell we obtain **μ**=2.52 V, and **σ**std=0.158 V, also smaller than the array variations. Based on these results, the variation of each cells is plotted on Figure 9(c) as a function of the average Vset of these individual cells. The challenges and responses of the PUF are generated after characterization of each cell, with a "0" state when Vset<μ, and a "1" state when Vset>μ.

4.2 *Statistical analysis – CRPs*

- The population of the ReRAM array is sorted in 8 multi-states 0 to 7 of equal probability 12.5%
- The resulting Vector Error is calculated in the "base" assuming a mean Vset of 2.1Volt
- When the distribution is drifting in a positive or negative direction VE is changing
- The profile of VEs is deterministic and can be anticipated by the Machine Learning Engine

Mi	Range Vset Volt	Median Vset Volt	Median σ Volt	VE_i % Neg Drift (1.8V)	VE_i % Neg Drift (1.95V)	VE_i % Base (2.1V)	VE_i % Pos Drift (2.25V)	VE_i % Pos Drift (2.4V)
0	1.466<	1.285	0.099	0	0	0	0	0
1	1.466-1.736	1.615	0.110	5	0.1	0	0	0
2	1.736-1.928	1.840	0.125	62	19	2	0.1	0
3	1.928-2.1	2.015	0.132	95	69	26	4	0.2
4	2.1-2.272	2.185	0.138	0.2	4	26	68	95
5	2.272-2.464	2.374	0.151	0	0.3	4	23	60
6	2.464-2.735	2.579	0.160	0	0	0	2	13
7	>2.735	2.928	0.175	0	0	0	0	0.2

Fig. 10A: Statistical analysis – Drifting distribution

Based on the experimental data presented in the previous section the drift between the Challenges and the responses is modelled assuming normal distributions. The analysis on the Fig 10A report impact of the drift of the Response on the CRP error rates, is by state. $\boldsymbol{VE_i}$ is computed by state from 0 to 7 for the base. When the Responses are drifting in a positive direction, respectively to 2.25V and 2.4V, the CRP error rates of the first four states are going down, while the CRP error rates of the last four states are going up. A reverse effect is observed for negative drifts to 1.95V and 1.8V. In Figure 10B the analysis is related to the respective change of the standard deviation of the entire population versus the standard variation of each cells.

- σ_D is the standard variation of the entire distribution of all cells – Base vs Tighter or Wider
- σ_{Mi} is the standard variation of the distribution of all cells that are described by the state Mi

Mi	σ_D/σ_{Mi} Tighter -20%	σ_D/σ_{Mi} Base ---	σ_D/σ_{Mi} Wider +20%	VE_i % Tighter (0.8σ_D)	VE_i % Base (σ_D)	VE_i % Wider (1.2σ_D)
0	4.54	5.45	6.83	0	0	0
1	4.09	4.91	6.14	0	0	0
2	3.60	4.32	5.40	4	2	0
3	3.42	4.09	5.09	30	26	21
4	3.25	3.91	4.91	30	26	22
5	2.98	3.58	4.46	7	4	1
6	2.81	3.37	4.21	1	0	0
7	2.57	3.09	3.86	0	0	0

Fig. 10B: Statistical analysis – Variable distribution

If the spread of the general population of Responses to the PUF is getting tighter compared with the spread of Responses to an individual cell the average error rates across the 8 states will go up.

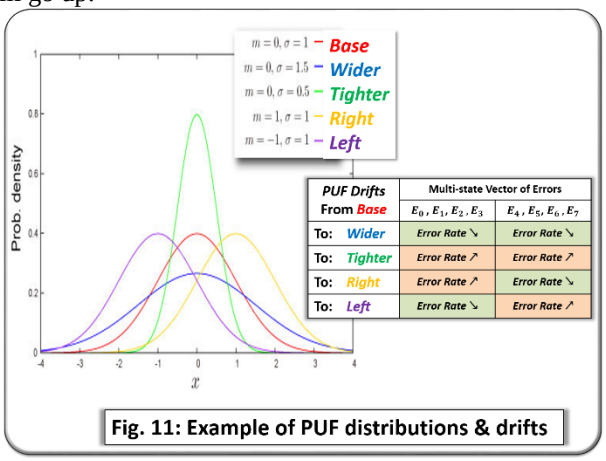


Fig. 11: Example of PUF distributions & drifts

Conversely if the spread is relatively wider, the average defect rates is going down. The analysis showing the impact of drifts of the physical parameters on the CRPs Vector of Errors VEs is summarized on Fig.11. The method to capture the profile of the physical parameters underlining a PUF with multi-states result in a predictable tracking of the drifts.

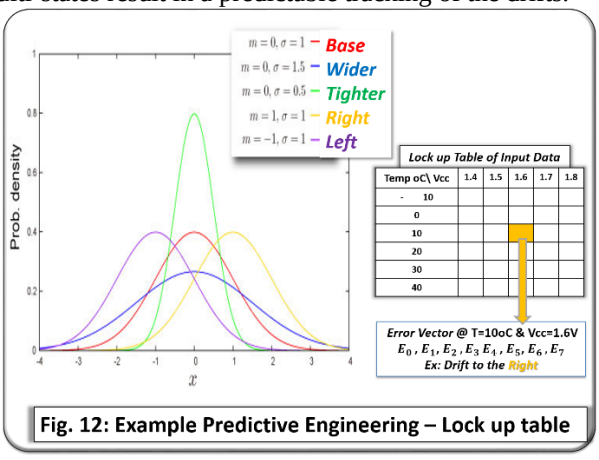


Fig. 12: Example Predictive Engineering – Lock up table

The effect of the external parameters such as temperature and bias conditions can enhance the accuracy of the computations of the MLE, see Fig 12. A set of lock up tables can be generated upfront with anticipated Vectors of Errors in each location, then stored in the working memory of the MLE, thereby further enhancing the accuracy of the authentication process. During a particular authentication cycle a sensed Vector of Input can allow the MLE to extract from the lock up table the expected Vector of Error as part as the "learning" data base.

5. SUMMARY

Security considerations: PUFs are strong cryptographic primitives because a fresh Response is generated by the hardware as often as needed to offer a secure, trusted authentication. Alterations to the PUF due to foreign intervention, or attempts to present a fake Challenge should be flagged by a negative authentication. The method described in this disclosure has the objective of enhancing the strength of PUFs by reducing the negative influence of natural drifts and variations of the physical parameters underlining the PUF. Other important factors in judging the strength of PUFs are their unclonability, and their ability to block foreign entity to access secret information, such as the PUF Challenge or the mapping of the multi-states. Ways to improve security when multi-state architecture is involved include:

- Storing the mapping of the multi-state during Challenge generation within an embedded secure memory, the Option A. Thereby mapping is only generated once, together with the Challenges. However correctly securing the secrecy of the storage is pivotal to the value of the method.
- Generating a fresh mapping of the multi-states during Response generation, and downloading it to the MLE, the Option B. In this case no data is stored, and post authentication there is no information left to be stolen by third party.
- Finally there is the possibility to send the mapping of the multi-states directly to the secure sever together with the Challenges. When the data transferred between the PUF and the secure server is encrypted, this method is also safe.

False Positives Authentication (FPA) versus False Negative Authentication (FNA): The physical parameters underlying PUFs as described in this paper follows the laws of physics that make them predictable when subject to effects such as temperature or bias changes. The methods described in this work are leveraging this predictability to reduce FNA without having a negative impact on FPA, the related error correction methods will not correct random defects. The usage of weaker physical parameters will have acceptable FNAs, while making side channel attacks more difficult.

Future work: Considering the encouraging statistical analysis done with ReRAM samples additional funding were secured to pursue the development of PUF components based on these methods. Short term we expect to model of the effect of temperature and biasing conditions on the Vset, and anticipated CRP error rates. We intend to develop several coding-decoding options, and analyze the respective benefits in reducing FNAs, and FPAs.

6. REFERRENCES

1. **D. Naccache and P. Frémanteau**; Unforgeable identification device, identification device reader and method of identification; *Patent US5434917; Aug. 1992;*
2. **Pravin Prabhu and all;** Extracting Device Fingerprints from Flash Memory by Exploiting Physical Variations; *4th int. conf. on Trust and trustworthy computing; June 2011;*
3. **Todd Alan Christensen, and all;** Implementing Physically Unclonable Function (PUF)utilizing EDRAM memory cell capacitance variation; *Patent No.: US 8,300,450 B2; Oct. 30, 2012*
4. **Zhu, S.M. Millendorf, and all;** Physically Unclonable Function based on resistivity of magnetoresistive random-access memory magnetic tunnel junctions; *Patents. US 2015/0071432 A1; March 2015;*
5. **Elena Ioana Vatajelu, Lionel Torres, and all;** STT-MRAM-Based PUF Architecture exploiting Magnetic Tunnel Junction Fabrication-Induced Variability; *ACM transactions; July 2015;*
6. **S. M. Van Rijnswou;** Device with a Secure Virtual Machine; *Patent publication US2010/0199104 A1, Aug 2010;*
7. **Z. Paral, S. Devadas**; Reliable PUF value generation by pattern matching; *Pub. US2012/0183135 A1, Jul 2012;*
8. **M.Rostami, and all;** PUF Authentication and Key-Exchange by Substring Matching; *Patent application US20150195088A1, Jan 2014;*
9. **M. Schmid, and all;** System and Method for defending against malicious software; *US patent No:7,085,928 B1, Aug 2006*;
10. **Marten E. Van Dijk;** System and Method of Reliable Forward Secret Key Sharing with Random Functions; *Patent publication US2008/0044027A1, Feb 2008*;
11. **Price, N. E., and A. T. Sherman;** How to Generate Repeatable Keys Using PUF, Correcting PUF Errors with Iteratively Broadening and Prioritized Search; *International Association for Cryptologic Research, CHES 2014*;
12. **Boehm, H. M.;** Error correction coding for PUF: *in Austrochip. Workshop in Microelectronics, 2010;*
13. **Yu, M., and S. Devadas;** Secure and Robust Error Correction for PUF; *IEEE Design & Test of Computers, Verifying Physical Trustworthiness of ICs and Systems, 2010;*
14. **Herder, C., and all;** "PUFs and Applications; *A Tutorial." Proceedings of the IEEE 102, no. 8 (2014): 1126-1141;*
15. **B. Gassend;** Physical random functions; *M.S. thesis, Dept. Electr. Eng. Comput. Sci., MA, USA, Massachusetts Inst. Technol., Cambridge,, 2003;*
16. **Guajardo, J, and all;** PUFs and PublicKey Crypto for FPGA IP Protection; *Field Programmable Logic and Applications, 2007.*
17. **Guajardo, J., and all;** FPGA Intrinsic PUFs and Their Use for IP Protection; *CHES, 2007;*
18. **Suh, G. E;** AEGIS: A single-chip secure processor; *Ph.D. dissertation, Dept. Electr. Eng. Comput. Sci., MIT., 2005;*
19. **Maes, and all;** A Soft Decision Helper Data Algorithm for SRAM PUFs; *2016 IEEE International Symposium on Information Theory. 2009;*
20. **C.Helfmeier, and all;** Physical Vulnerabilities of PUFs; *Proceedings of the conference on Design, Automation & Test (DARE'14). Belgium, 2014;*
21. **Helfmeier, and all;** PUFs: Myth, Fact or Busted? A Security Evaluation of Physically Unclonable Functions Cast in Silicon; CHES, 2012.
22. **Hvarinen, and Oja;** Independent Component Analysis, algorithms, and applications; *Neural Networks, Vol 13, 411-430, 2000;*
23. **Llyod, S;** Least Squares Quantization in PCM*; IEEE Transactions on Information Theory 18, no. 2 (1982);*
24. **Yang, L, and D. and Wu;** Adaptive Quantization Using Piecewise Companding and Scaling for Gaussian Mixture; *Visual Communication and Image Representation. 2012;*
25. **Berrou, C., and all**; Near Optimum Error Correcting Coding and Decoding: Turbo-Codes; *IEEE Trans. on Communications 44, no. 10 (1996): 1261-1271;*
26. **Razi, A., and all;** Binary Source Estimation Using a Two-Tiered Wireless Sensor Network; *Communications Letters, IEEE 15, no. 4 (2011): 449-451;*
27. **Razi, A., M. Ardebilipour, and F., Afghah;** Space-Time Block Codes Assisted by Fast Turbo Codes; *WiCOM '08. 2008. 1-6;*
28. **Afghah, Fatemeh, Mehrdad Ardebilipour, and Abolfazl Razi**; Concatenation of space-time block codes and LDPC codes. *The 13th INTNSPS, 2008. 1-5;*
29. **Dodis, and all;** Fuzzy Extractors: How to Generate Strong Keys from Biometrics and Other Noisy Data; *EUROCRYPT, 2004: 523-540;*