Modeling Strong Physically Unclonable Functions with Metaheuristics

Carlos Coello Coello
CINVESTAV-IPN, Departamento de Computación
Mexico City, Mexico
ccoello@cs.cinvestav.mx

Marko Durasevic
University of Zagreb, Faculty of Electrical Engineering and Computing
Zagreb, Croatia
marko.durasevic@fer.hr

Domagoj Jakobovic
University of Zagreb, Faculty of Electrical Engineering and Computing
Zagreb, Croatia
domagoj.jakobovic@fer.hr

Marina Krcek
Delft University of Technology
Delft, The Netherlands
m.krcek@tudelft.nl

Luca Mariot
University of Twente
Enschede, The Netherlands
l.mariot@utwente.nl

Stjepan Picek
Digital Security Group, Radboud University
Nijmegen, The Netherlands
stjepan.picek@ru.nl

ABSTRACT

Evolutionary algorithms have been successfully applied to attack Physically Unclonable Functions (PUFs). CMA-ES is recognized as the most powerful option for a type of attack called the reliability attack. In this paper, we take a step back and systematically evaluate several metaheuristics for the challenge-response pair-based attack on strong PUFs. Our results confirm that CMA-ES has the best performance, but we note several other algorithms with similar performance while having smaller computational costs.

CCS CONCEPTS

• **Mathematics of computing** → **Evolutionary algorithms**; • **Security and privacy** → *Hardware attacks and countermeasures.*

KEYWORDS

Metaheuristics, Physically Unclonable Functions, CMA-ES, CRP

ACM Reference Format:
Carlos Coello Coello, Marko Durasevic, Domagoj Jakobovic, Marina Krcek, Luca Mariot, and Stjepan Picek. 2023. Modeling Strong Physically Unclonable Functions with Metaheuristics. In *Genetic and Evolutionary Computation Conference Companion (GECCO '23 Companion), July 15–19, 2023, Lisbon, Portugal.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3583133.3590699

1 INTRODUCTION

Physically Unclonable Functions (PUFs) are (partly) disordered physical systems that can be challenged with external stimuli upon which they react with the corresponding responses. Those responses will depend on the nanoscale structural disorder present in the PUFs. When supplied with the same challenge, no two PUFs will give the same response. While the name suggests that a PUF cannot be cloned, numerous results have shown that various artificial intelligence techniques could easily model its behavior. Rührmair et al. provided the first results showing it is possible to model PUFs [15]. The authors evaluate "various machine learning techniques, including Logistic Regression and Evolution Strategies." They showed that both techniques work well, but with evolution strategies, they managed to break specific problem instances that were not breakable with logistic regression. In the following years, the Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [9] became the dominant option for attacking PUFs with EAs, especially concerning the reliability attack. For instance, G. Becker used CMA-ES to break a specific type of PUF called XOR PUF, where the author broke commercial PUF-based RFID tags [1]. Interestingly, while CMA-ES performed well, there was little (or no) research investigating whether other metaheuristics perform comparably or how difficult it is to tune such algorithms.

2 PHYSICALLY UNCLONABLE FUNCTIONS

PUFs use inherent manufacturing differences within every physical object to give each instance a unique identity. A PUF can be queried with challenges to receive a number of responses (challenge-response pairs - CRP).

Arbiter PUFs (APUFs) consist of one or more chains of 2-bit multiplexers pairs with identical layouts. Each multiplexer pair is denoted as a stage, with n stages in a single chain. A single input signal is introduced to the first stage to both the bottom and top multiplexer in the pair. The chain is fed a control signal of n bits called a challenge, where each bit determines whether the two input signals in that stage would be switched (crossed over by the multiplexer) or not. In ideal conditions, the input signal would propagate at the same speed through each stage, and both the lower and upper signals would arrive at the arbiter (at the end of the chain) simultaneously. Due to manufacturing inconsistencies, each multiplexer delay is slightly different, and the top and bottom input signals are not synchronized. At the end of the chain, the arbiter determines which signal arrived earlier and forms the response (0 or 1). The response of a PUF is determined by the delay difference between the top and bottom input signal, which is, in turn, the sum of delay differences of the individual stages. An APUF with

n challenges consists of $n + 1$ stages, as the last stage is due to the arbiter.

To increase the resistance of APUFs against machine learning attacks, it is possible to add nonlinear elements to the PUF design. One standard method to do so is the so-called XOR APUF design [17]. In a k-XOR APUF, k Arbiter PUFs are placed on the chip. Each of the Arbiter PUFs receives the same challenges, and the responses of the k PUFs are XORed to build the final response bits.

CRP-based Attack. When attacking a PUF, it is often not necessary to reach very high accuracy, and any prediction accuracy significantly higher than 50% can be considered a successful attack [6]. To efficiently model a PUF, one tries to determine the delay vector $w = (w_1, \dots, w_{n+1})$ that models the delay differences in each stage. Lim et al. proposed a linear additive model that captures the APUF behavior where we require the map $f(c) = \phi$ of the applied challenge c of length n to a feature vector ϕ of length $n+1$ [14]. The product of the feature vector and delay vector decides which signal came first, and based on this, what is the response bit r:

$$\phi_i = \prod_{l=i}^{k}(-1)^{c_l}, \text{for } 1 \leq i \leq k. \tag{1}$$

$$r = \begin{cases} 1 & \text{if } \vec{w}\vec{\phi}^T < 0 \\ 0 & \text{if } \vec{w}\vec{\phi}^T > 0. \end{cases} \tag{2}$$

The research community explored various algorithms for modeling attacks, both from the EA and machine learning domains. For more details, we refer interested readers to [5].

Reliability Attack. Due to added nonlinearity, attacking XOR APUFs is more difficult than attacking APUFs. To make the attack easier, it is also possible to consider the reliability of a response, i.e., how often the PUF evaluates to the same response bit for a given challenge [7].[1] The main idea of the reliability attack is to make repeated measurements for the same challenge and observe which response bits are stable and which sometimes flip. Then, if the response for a given challenge is unstable, it is likely that the corresponding delay difference is close to zero [1].

3 METHODOLOGY

In a CRP-based attack, the attacker tries to infer the delay vector that adequately describes the PUF under consideration. The obvious choice for the solution encoding is a floating-point vector that models the target device behavior. For an XOR APUF will have $k \times (n + 1)$ floating-point values.

From the attacker's perspective, the optimization goal is to predict as many correct responses as possible, given a set of challenges. The model optimization is performed on the learning set of CRPs, while the generalization capability is evaluated on a separate test set. Therefore, our simple fitness function minimizes the number of errors, that is, wrongly predicted responses:

$$fitness = number\ of\ prediction\ errors. \tag{3}$$

[1]While using reliability, the attack in [7] did not apply it to XOR APUFs.

The attacks are simulated on a number of PUF sizes in increasing complexity. Therefore XOR APUFs of sizes 4×16, 4×32, and 4×64 are considered, which are commonly found in available devices.

The *learning set* for every PUF size is devised in the following manner: we randomly create ten independent PUFs and generate corresponding learning sets with varying numbers of CRPs, ranging from 2 000 to 250 000. Every algorithm is applied to each of the ten instances in five runs, which results in 50 independent runs. The reason for using ten different PUF instances is to reduce the bias that could arise from experimenting on a single PUF only. All the PUFs are instantiated by randomly generating their delay vectors using a normal distribution with parameters $\mathcal{N}(0, 1)$, following the existing literature. The test set is produced with the same PUF instances used in the learning phase but with different challenge-response pairs. Each PUF instance will have a single test set, and solutions obtained on all five runs will be tested on that same set. Unlike the learning sets, all the test sets have the same size of 1 000 CRPs.

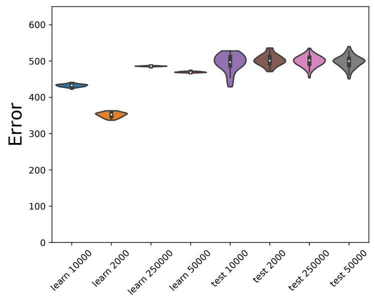
We applied the following algorithms to this problem: artificial immune system algorithm (AIS) [2], clonal selection algorithm (CLONALG) [4], covariance matrix adaptation ES (CMA-ES) [10], differential evolution (DE) [16], a generational genetic algorithm with roulette wheel selection (RW), and a steady-state genetic algorithm with tournament selection (SST) [8]. Besides these algorithms, evolution strategy (ES) [3], particle swarm optimization (PSO) [13], artificial bee colony (ABC) [12], and elimination genetic algorithm [11] were also tested. However, due to their poor performance during the parameter tuning procedure, they were not considered in further experiments. The algorithmic descriptions are omitted to save space, but their definitions and available parameters can be found on the used library website: http://ecf.zemris.fer.hr/.

4 EXPERIMENTAL RESULTS

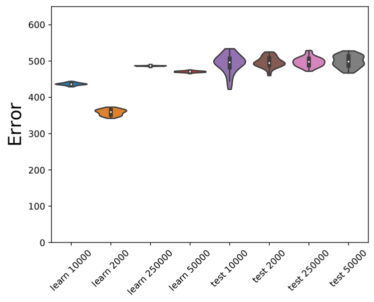
Table 1 shows the results obtained for different XOR APUF instances. It is clear that even for the simplest instance, many algorithms have trouble achieving a low error rate. In many cases, the algorithms perform only slightly better than random estimation. A larger number of CRPs with these problem instances is necessary to achieve better results. However, even this is not enough to guarantee good performance; this is best seen with the 4×64 instance, where only CMA-ES obtained acceptable accuracy.

Figure 1 shows the violin plots of the results obtained for the XOR APUF instance (4×64). The figures show that all algorithms except CMA-ES achieve poor results. In all cases the performance on the learning set deteriorates as the number of CRPs increases, but the performance on the test set usually remains similar. This indicates that it is more difficult for the algorithms to learn the correct configuration from the given instances and that the results on the training set are misleading if not enough CRPs are used.

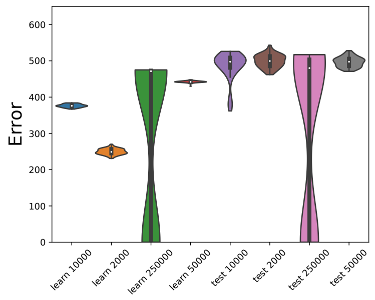
To investigate the correlation between training and test sets' performance, we select CMA-ES and calculate the correlation between its performance in the training and test sets using Spearman's rank correlation coefficient. The results of these tests are shown in Table 2. The table shows almost no correlation on the smallest learning sets for all PUF instances. However, as the number of CRPs increases, so does the correlation.



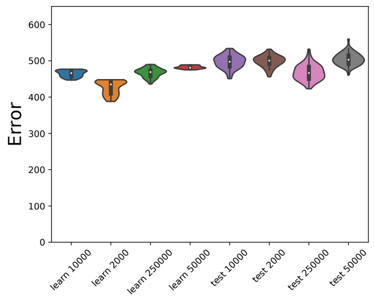
(a) AIS.



(b) CLONALG.

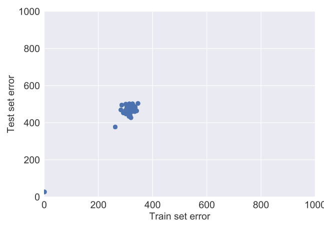


(c) CMA-ES.

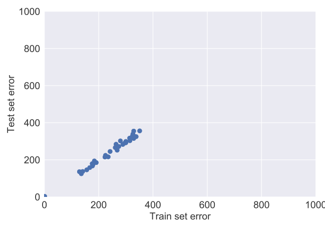


(d) DE.

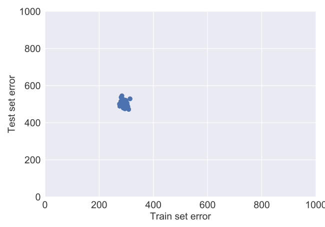
Figure 1: Violin plots of the results obtained for each algorithm across all executions for 4×64 XOR APUFs.



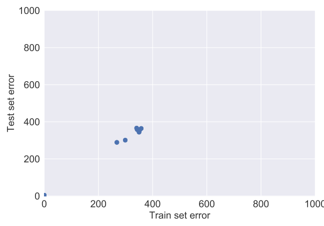
(a) 4×16, 2 000 CRP.



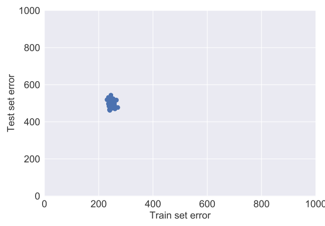
(b) 4×16, 250 000 CRP.



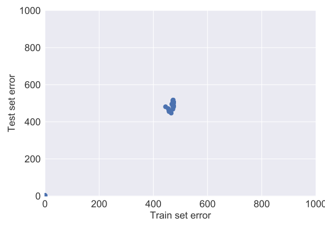
(c) 4×32, 2 000 CRP.



(d) 4×32, 250 000 CRP.



(e) 4×64, 2 000 CRP.



(f) 4×64, 250 000 CRP.

Figure 2: Scatter plots outlining different correlation levels for CMA-ES and various XOR APUF instances.

Figure 2 shows scatter plots for CMA-ES when optimizing different PUF instances and using the smallest and largest number of CRPs. The x-axis represents the error obtained on the training set, while the y-axis represents the error obtained on the test set. In the case of the 4×16 instance, it can be observed that for 2 000 CRP, there is little correlation between the results obtained on both sets. When the number of CRPs increases, a much better correlation can be achieved. For the 4×64 instance, it should be noted that, although the images look similar, the results are much better correlated for the larger number of CRPs, due to a number of runs obtaining a very low error, all concentrated in the lower-left corner of the plot.

Table 1: Lowest errors obtained by different XOR APUFs on the test set.

PUF	CRP	Method					
		AIS	CLONALG	CMA-ES	DE	RW	SST
4×16	2 000	468	460	27	222	409	463
	10 000	451	456	0	162	213	28
	50 000	166	14	0	131	160	9
	250 000	32	13	0	139	165	4
4×32	2 000	470	466	472	468	474	469
	10 000	457	474	455	312	435	464
	50 000	463	478	0	292	345	471
	250 000	200	52	0	266	341	11
4×64	2 000	471	460	462	456	472	459
	10 000	429	422	362	451	431	427
	50 000	451	467	471	461	475	470
	250 000	454	472	0	423	469	468

Table 2: Results of the Spearman test for the CMA-ES algorithm.

PUF	CRP			
	2 000	10 000	50 000	250 000
4×16	0.157	0.961	0.962	0.989
4×32	-0.201	0.396	0.836	0.764
4×64	-0.026	0.224	0.016	0.803

We hypothesize that CMA-ES is a more successful algorithm as it has adaptive parameters that can respond to the dynamics of the convergence process. Still, once the problem instances become too large, even CMA-ES cannot adapt sufficiently to benefit from the adaptive parameters unless we use many CRPs.

The performance of the algorithms is closely related to the number of CRPs. For APUFs, the algorithms can perform quite well even with a relatively small number (e.g., 2 000 CRPs). However, as soon as XOR APUFs are considered, such a number of CRPs - except for the simplest problem instance - is no longer sufficient to achieve satisfactory results. Even more, if too small a number of CRPs is used, the algorithms achieve performance that is slightly better than random estimation. Therefore, it is of utmost importance to use a sufficiently large number of CRPs in training sets to obtain configurations that generalize well.

While CMA-ES performs the best in general, based on our results, we would not recommend this algorithm for APUF cases where the number of CRPs is low as we see overfitting and the runtime is not favorable compared to other tested algorithms. Instead, we recommend using the RW algorithm in such cases. FOR XOR APUF, there does not seem to be an acceptable alternative to CMA-ES among the tested algorithms. Still, even CMA-ES struggles with more difficult problem instances. We recommend using as large as possible number of CRPs but also test the algorithm performance multiple times as we notice the performance varies significantly.

5 CONCLUSIONS

We observe that the main condition for a successful attack is the size of the training set. What is more, there is a correlation between the performance on the training and test sets: if the performance on the training set is good, it will also be good for the test set.

ACKNOWLEDGMENTS

Carlos A. Coello Coello gratefully acknowledges support from CONACyT grant no. 2016-01-1920 (Investigacion en Fronteras de la Ciencia 2016).

REFERENCES

[1] Georg T. Becker. 2015. The Gap Between Promise and Reality: On the Insecurity of XOR Arbiter PUFs. In *CHES 2015, Proceedings (LNCS, Vol. 9293)*, Tim Güneysu and Helena Handschuh (Eds.). Springer, 535–555.

[2] Heder S. Bernardino and Helio J. C. Barbosa. 2009. Artificial Immune Systems for Optimization. In *Nature-Inspired Algorithms for Optimisation*. Springer Berlin Heidelberg, 389–411. https://doi.org/10.1007/978-3-642-00267-0_14

[3] Hans-Georg Beyer and Hans-Paul Schwefel. 2002. Evolution strategies - A comprehensive introduction. *Nat. Comput.* 1, 1 (2002), 3–52.

[4] L.N. de Castro and F.J. Von Zuben. 2002. Learning and optimization using the clonal selection principle. *IEEE Transactions on Evolutionary Computation* 6, 3 (2002), 239–251. https://doi.org/10.1109/TEVC.2002.1011539

[5] Jeroen Delvaux. 2017. *Security Analysis of PUF-based Key Generation and Entity Authentication ; Veiligheidsanalyse van PUF-gebaseerde sleutelgeneratie en entiteitsauthenticatie.* Ph. D. Dissertation. Katholieke Universiteit Leuven, Belgium. https://lirias.kuleuven.be/handle/123456789/581770

[6] Jeroen Delvaux. 2019. Machine-Learning Attacks on PolyPUFs, OB-PUFs, RPUFs, LHS-PUFs, and PUF-FSMs. *IEEE Trans. Inf. Forensics Secur.* 14, 8 (2019), 2043–2058.

[7] Jeroen Delvaux and Ingrid Verbauwhede. 2013. Side channel modeling attacks on 65nm arbiter PUFs exploiting CMOS device noise. In *2013 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*. 137–142. https://doi.org/10.1109/HST.2013.6581579

[8] David E. Goldberg. 1989. *Genetic Algorithms in Search, Optimization and Machine Learning* (1st ed.). Addison-Wesley Longman Publishing Co., Inc., USA.

[9] Nikolaus Hansen. 2006. The CMA Evolution Strategy: A Comparing Review. In *Towards a New Evolutionary Computation - Advances in the Estimation of Distribution Algorithms*, José Antonio Lozano, Pedro Larrañaga, Iñaki Inza, and Endika Bengoetxea (Eds.). Studies in Fuzziness and Soft Computing, Vol. 192. Springer, 75–102.

[10] N. Hansen and S. Kern. 2004. Evaluating the CMA Evolution Strategy on Multimodal Test Functions. In *Parallel Problem Solving from Nature PPSN VIII (LNCS, Vol. 3242)*, X. Yao et al. (Eds.). Springer, 282–291.

[11] D Jakobovic. 1997. Adaptive genetic operators in elimination genetic algorithm. In *Proceedings of the 19th International Conference on Information Technology Interfaces*. 351–356.

[12] Dervis Karaboga et al. 2005. *An idea based on honey bee swarm for numerical optimization.* Technical Report. Technical report-tr06, Erciyes university, engineering faculty, computer ....

[13] J. Kennedy and R. Eberhart. 1995. Particle swarm optimization. In *Proceedings of ICNN'95 - International Conference on Neural Networks*, Vol. 4. 1942–1948 vol.4. https://doi.org/10.1109/ICNN.1995.488968

[14] Daihyun Lim, Jae W. Lee, Blaise Gassend, G. Edward Suh, Marten van Dijk, and Srinivas Devadas. 2005. Extracting secret keys from integrated circuits. *IEEE Trans. Very Large Scale Integr. Syst.* 13, 10 (2005), 1200–1205.

[15] Ulrich Rührmair, Frank Sehnke, Jan Sölter, Gideon Dror, Srinivas Devadas, and Jürgen Schmidhuber. 2010. Modeling Attacks on Physical Unclonable Functions. In *Proceedings of the 17th ACM Conference on Computer and Communications Security* (Chicago, Illinois, USA) *(CCS '10)*. Association for Computing Machinery, New York, NY, USA, 237–249. https://doi.org/10.1145/1866307.1866335

[16] Rainer Storn and Kenneth V. Price. 1997. Differential Evolution - A Simple and Efficient Heuristic for global Optimization over Continuous Spaces. *J. Glob. Optim.* 11, 4 (1997), 341–359.

[17] G. Edward Suh and Srinivas Devadas. 2007. Physical Unclonable Functions for Device Authentication and Secret Key Generation. In *2007 44th ACM/IEEE Design Automation Conference*. 9–14.