Elaborating on Sub-Space Modeling as an Enrollment Solution for Strong PUF

Amir Ali Pour, David Hély, Vincent Beroulle, Giorgio Di Natale

Elaborating on Sub-Space Modeling as an Enrollment Solution for Strong PUF

Amir Ali-pour*, David Hely†, Vincent Beroulle‡ and Giorgio Di Natale§
(*, ‡) Univ. Grenoble Alpes, Grenoble INP, LCIS, 26000 Valence, France
(†) Univ. Grenoble Alpes, CEA, LETI, F-38000 Grenoble, France
§ Univ. Grenoble Alpes, CNRS, Grenoble INP, TIMA, 38000 Grenoble, France,
(*, ‡, §) Email: firstname.lastname@univ-grenoble-alpes.fr

***Abstract*—In this work we present sub-space modeling of strong PUF as a cost efficient solution for PUF enrollment for the designers' community. Our goal is to demonstrate a method which can reduce the overall cost in terms of number of CRPs required for training, training time and memory. Instead of modifying the estimated model structure, we propose to reduce the complexity of the modeling target. This means to provide secured access to the internal responses of strong PUF during the enrollment and capture internal CRPs to model each sub-component of the PUF independently. It also necessitates to permanently remove the internal access after the enrollment to prevent exposure of the internal responses. This means that the internal responses should not be directly accessible after enrollment. Our sub-space modeling method requires lesser number of CRPs compared to modeling the whole PUF. We experimentally prove that sub-space modeling can significantly reduce the cost of training compared to some of the latest works. For instance, we could model 128-stage 6-XOR Arbiter PUF with just above 90% prediction accuracy with 5000 CRPs. Here the response in the CRP is a vector including the responses of the sub-components. Our results show that sub-space modeling is potentially a cost-efficient solution to enroll strong PUF with high complexity.**

***Index Terms*—Physically Unclonable Function (PUF), XOR Arbiter PUF, PUF Enrollment, Machine Learning (ML), Logistic Regression (LR)**

I. Introduction

Physically Unclonable Functions (PUFs) are considered nowadays as one of the emerging security primitives for resource-constraint ecosystems in the field of IoT [1]. PUFs are pervasively used to generate device-specific data which can be used in several applications such as light-weight device authentication, and encryption key generation [2]. PUF is characterized as a hardware bound function which utilizes the unit-specific micro-variations to generate device-specific data. The functionality of PUF is based on mapping a bit-vector challenge (the input) to a response (output) and generate a so called Challenge-Response-Pair (CRP). Variations of PUF structure exist, which differ in how the PUF functions and how many device-specific data the PUF can generate.

Strong PUF is a macro variant which aims at generating an abundance of device-specific identifiers. Very large number of CRPs can be generated by an strong PUF depending on the size of the challenge, so called the input dimensiality of the PUF [3]. This characteristic in turn makes strong PUF an ideal primitive for cryptographic applications such as single-use (one-time pad; OTP) key generation [4], [5].

Usually strong PUF is enrolled via a database of CRPs captured from the PUF circuit before deployment [2], [4]. However, several authentication and key generation protocols exist which suggest enrolling PUF with its software model equivalent, or an estimation model which can identify the PUF [6]–[9]. In the primary approach, a software model is the stored data on the verifier server to provide access to the full CRP space of the enrolled strong PUF. This in turn solves the shortage of CRPs on the verifier server, eliminating the requirement to re-enroll and thus restock on the CRP database. It also can be considered as a compact solution for enrolling the PUF, therefore, requiring less memory space on the storage device of the verifier server [9].

The common approach in building the equivalent software model of PUF is to use Machine Learning (ML) modeling techniques. The idea is to train a probabilistic model which can estimate the CRP characteristic of the PUF with high probability. In this approach, a set of CRPs is captured and a training algorithm is used to converge the model's characteristic to an estimation of the target PUF's CRP characteristic, according to the captured CRP set [10], [11].

An important challenge in ML-based modeling of PUF is to deal with the structural complexity of the PUF (for instance k-XOR Arbiter PUF with k larger than 4). Usually the strong PUF with high complexity require significantly large number of CRPs for training [12]–[17]. This on one hand is appealing for the designers' community to implement strong PUF with high complexity to protect against model building attacks. On the other hand, it imposes additional cost for protocols which rely on enrolling the PUF with ML-based modeling. This is due to the fact that CRP collection is done usually during the manufacturing test phase, since doing it post fabrication is an expensive operation. Therefore, spending more time during the test phase to collect large number of CRPs, increases the time of testing and consequently the manufacturing cost. In addition, large CRP training set size leads to spending hours of training time per PUF model. This in turn leads to excessive computation power usage and thus increasing the cost of modeling. These factors generally imply that modeling

This material is based upon the work supported by the French National Research Agency in the framework of the "Investissements d'avenir" program (ANR-15-IDEX-02).

strong PUF with high complexity using the conventional ML methods is an expensive solution for enrollment.

This work is inspired to put sub-space modeling into practice as a cost-efficient solution for enrollment. In sub-space modeling, the assumption is that the designer can access the internal values of strong PUF with large complexity during the test phase. In this way, the designer has multiple modeling targets with reduced complexity, which in turn need fewer CRPs for training compared to the whole PUF. In this work, we show how sub-space modeling can be performed on XOR Arbiter PUF to provide a model with a significantly reduced cost. Our contributions will be to develop an ML-based enrollment solution with the following features:

- Able to generate (at server level) all the possible CRPs of the target PUF.
- Requiring a small amount of memory at server level to store such an information.
- Providing a constant and short enrollment time per PUF, thus applicable in a real industrial/commercial environment.

In turn, sub-space modeling can be a suitable enrollment solution for the designers community compared to other conventional modeling methods. Given that with ML-based enrollment, access to the full CRP space is provided which can emerge into new protocols for authentication and encryption key generation. Such protocols require also novel approaches to restrict CRP access after enrollment. This is crucial, as it prevents openly accessible CRPs to all parties after enrollment to avoid model-building attacks. Moreover, it is important that the physical access to the I/O PUF is disabled once the PUF is enrolled successfully. An example of that can be found in [6], where it is suggested to permanently disable the physical access-points to the PUF, e.g., by burning irreversible fuses so that other parties cannot access directly the PUF.

The structure of our paper is the following. In section II we present the preliminaries to PUF and modeling PUF for enrollment with Machine Learning (ML). In section III explains our proposed method. In section IV we describe our evaluation setup and later present the experimental works and the comparisons. In section V presents the conclusion of our work.

II. Preliminaries

A. Xor Arbiter PUF

One of the known strong PUF structures is the XOR Arbiter PUF. Arbiter PUF was first introduced in 2002 by Gassend et al in [18]. The idea of Arbiter PUF is based on the delay difference between two racing paths which are structurally similar, but due to minor process variations, differ in time of passing a signal given to them at the same time. The structure of XOR Arbiter PUF is based on multiple Arbiter PUFs whose input (challenge) are of the same size, and triggered by a global input. The output of the XOR Arbiter PUF is also the XOR of the output of each Arbiter PUF. Fig. 1 shows the structure of an n-stage k-XOR XOR Arbiter PUF.

As shown in Fig. 1, the structure of a n-stage k-XOR Arbiter PUF can be divided into k n-stage Arbiter PUFs with independent responses. Thus each n-stage Arbiter PUF can be considered a sub-component of the main XOR Arbiter PUF. Such architecture then can benefit from sub-space modeling. Accordingly, each n-stage Arbiter PUF can be modelled independently, and merged after to build the model of the whole n-stage k-XOR Arbiter PUF.

B. Strong PUF Enrollment with ML

The goal of PUF enrollment with ML-based modeling is to replace the conventional CRP database with an estimated model of the PUF. Let us denote the estimated model of PUF as h_{PUF}. The enrollment of PUF with ML-based modeling means that the verifier server will own h_{PUF} which provides access to the full CRP space of the PUF circuit with some miss-prediction error which is tolerable. During the enrollment the server has open access to the PUF CRPs to build the h_{PUF} (see Fig. 2 (a)). Let us consider c_i as a challenge input to the PUF circuit. If we observe the PUF circuit as a function f_{PUF} of c_i, then it's estimation can be defined as a function g_{PUF} of c_i and a set of internal values θ of the model. thus $h_{PUF} = \{g_{PUF}, \theta\}$. The estimation should then follow (1):

$$f_{PUF}(c_i) = r_i \approx r'_i = g_{PUF}(c_i, \theta) = h_{PUF}(c_i) \quad (1)$$

Where r_i is the PUF circuit's response to the challenge c_i and r'_i is the estimated model's prediction of r_i for c_i. The model then goes through an iterative training phase, where a learning algorithm modifies the internal values with respect to the CRP set and the function g_{PUF}.

We also assume that once the PUF is enrolled via its corresponding h_{PUF} model, the PUF then is protected with a masking protocol to provide a secure communication channel in mission mode that ensures adversaries won't be able to build easily an accurate model of the PUF (See Fig. 2 (b)).

At the beginning of the training phase, model h_{PUF} has a significant probability of erroneous estimation of the PUF's CRP characteristic. Therefore, the training runs iteratively until the probability of erroneous estimation is converged to zero or an acceptable minimum value. Since modeling here is done for the enrollment, we define metrics which are important for the enrollment, and we use them to evaluate the cost of training and the performance of the estimated models:

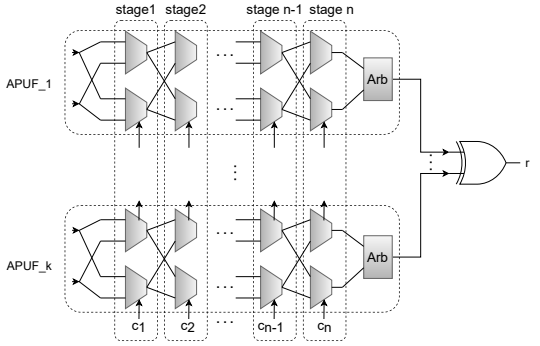

Fig. 1: The structure of n-stage k-XOR Arbiter PUF

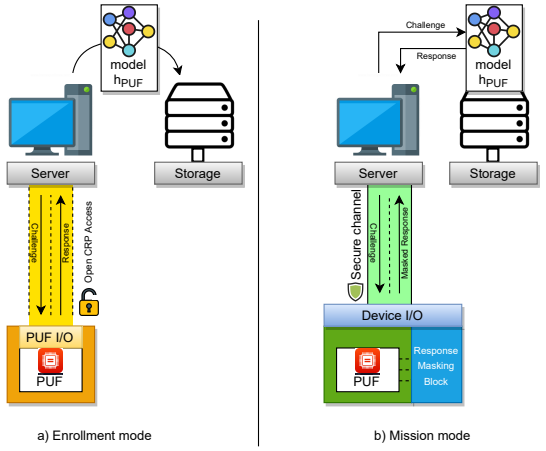

Fig. 2: Schematic of a communication between PUF and a verifier server, via an estimation model of PUF.

- **Prediction Accuracy** (ϵ)**:** Proportion of correctly predicted responses to total number of predictions.
- **Enrollment CRP Set Size** (css)**:** Size (in bytes) of the CRP set collected to enroll a given PUF circuit.
- **Total Time of Training** (T)**:** Time of training until an estimated model is generated with acceptable accuracy.
- **Estimated Model Size** (ms)**:** A measure of size (in bytes) of the internal trainable parameters θ of an estimated model of a PUF.

III. Sub-Space Modeling Method

A schematic of our proposed method is shown in Fig. 3. Here the illustration shows sub-space modeling for a variant of XOR Arbiter PUF. As shown in Fig. 3, the internal data from each sub-component ($r1$ to rk), in conjunction to their corresponding challenge values (e.g. $\{c1$ ... $ck\}$ + $\{r1\}$ for APUF_1, $\{c1$ ... $ck\}$ + $\{r2\}$ for APUF_2, etc.) are fed separately to trainer functions to discretely generate estimation models of each sub-component. After the trainer functions provide accurate estimated models of each sub-component, we merge the sub-models into forming a whole model which represents the whole PUF. We also take the assumptions below on how we can provide data for training.

- The strong PUF structure should be dividable into smaller sub-components (see Fig. 3) with reduced complexity. We assume that these sub-components are themselves functions of the input challenge to the PUF.
- An accompanying hardware extractor should be provided which has physical access to the internal sub-components' I/O (see Fig. 3). The extractor can capture the value of the internal sub-components in addition to the response of the whole-PUF, for any given challenge.
- We represent CR^vP for samples taken from PUF and its internal data. A CR^vP comprises a Challenge vector, and Response vector (R^v) which is a vector of values comprising the responses of internal sub-components as well as the response of the whole PUF.
- Number of sub-components addressed by the extractor are enumerable. We assume that the connectivity of the internal values to the extractor does not disturb the design and functionality of the PUF itself.
- Once the enrollment is successful and the accurate model is stored, the physical access to the internal values within the PUF circuit is permanently removed. This is essential to prevent future threads which may be able to regain access to the internal values via the extractor.

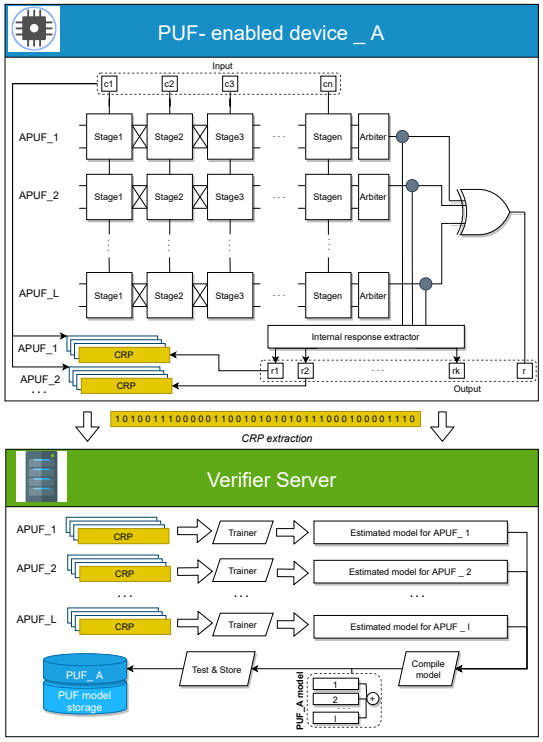

Fig. 3: Showing how sub-space modeling can be used for strong PUF modeling with separable components. Here the PUF variant is a n-stage k-XOR Arbiter PUF.

We also measure the accuracy of the model before storing it on sever. We compare the model's predicted responses with the puf responses for a set of challenges in a CRP set dedicated for testing (different from the ones used for training). After testing the model's prediction accuracy, it is stored for the given strong PUF on the verifier server for future use.

IV. Evaluation Setup and Experimental Results

In this section we evaluate our sub-space modeling with simulated CRP datasets of variants of XOR Arbiter PUF.

A. Experimental Setup

In our evaluation, we used a python based simulator of XOR Arbiter PUF [20]. This simulator has been used already

TABLE I: LR training hyper-parameters

Inverse regularization (C)	**Tolerance for stopping (tol)**	**Max iter**	**Solver**
1.0	0.0000001	10000	liblinear

TABLE II: A comparison of cost of training in modeling variants of XOR Arbiter PUF. Here SS refers to our proposed sub-space modeling method. R is the modeling method used in [12], T is the modeling method used in [14], M* is the modeling method first proposed in [19] and then revisited in [17]. S also is the modeling method used in [15].

	XOR size															
	4				5				6				7			
	css(MB)	ϵ	T(h)	ms(KB)	css(MB)	ϵ	T(h)	ms(KB)	css(MB)	ϵ	T(h)	ms(KB)	css(MB)	ϵ	T(h)	ms(KB)
R	0.377	99%	2:52	4.1	7.9	99%	16:36	5.1	N/A				NA			
T	3.1	98%	0:02	4.1	34.6	98%	00:12	5.1	236.2	98%	4:45	6.1	629.8	98%	66:53	7.2
M*	15.7	95%	<0:01	10.6	15.7	95%	<0:01	25.3	157.4	95%	<0:01	67.1	472.4	95%	0:02	199.7
S	0.944	98%	0:05	180.2	6.3	97%	1:5	1280	18.9	97%	6:1	3072	44.1	96%	18:2	10752
SS	**0.490**	**98%**	**<0:01**	**4.1**	**1.48**	**97%**	**<0:01**	**5.1**	**1.16**	**97%**	**<0:01**	**6.1**	**1.5**	**96%**	**<0:01**	**7.2**

in Ruhrmair's work in [12]. In this simulation, a n-stage k-XOR Arbiter PUF is simulated as a XOR function of k n-stage Arbiter PUFs. n-stage Arbiter itself is simulated as the sum of the signal propagation delays in each stage. The delay values in the simulator are generated randomly with a standard normal distribution, with mean 0 and standard deviation 1. During the instantiation of a PUF instance, the delay values are generated and allocated to the instance model. The PUF instance model is then ready to get a challenge vector and generate the corresponding responses.

We used the simulator to generate 10 instances of 128-stage $\{2, 3, 4, 5, 6, 7, 8, 9$ and $10\}$-XOR Arbiter PUF variants. For each variant we generated $100,000$ CR^vP, therefore a total of 9 Million CR^vPs for all the instances of all the variants have been generated. Noting that in this simulation we did not model the PUF instability noise explicitly. Therefore the randomness is only due to the signal propagation delay as we discussed earlier.

For modeling we used Sklearn's Logistic Regression (LR) to model each sub-component independently. The reason we chose LR is that comparing to other modeling techniques, LR shows to converge considerably faster, using less computation power. Also LR seems to be a good starting point to explore modeling techniques due to the fact that LR is relatively the simplest modeling technique compared to others such as Artificial Neural Networks or Support Vector Machine. In terms of the training specifications and hyper parameters, our entire parametric consideration for training with LR are given in Table I.

Noting in addition, that we measure css in terms of bytes for n-stage k-XOR Arbiter PUF as in (2), where N is the number of CRPs in a given training set.

$$css_{(byte)} = \frac{N \times (n + k + 1)}{8} \tag{2}$$

B. Experimental Results and Discussions

First we measure the cost of training using sub-space modeling, and compare it to some of the known and recent modeling methods. TableII shows the cost of training with respect to various k in n-stage k-XOR Arbiter PUF. Here we compare our sub-space modeling method (SS), to other modeling methods practiced for XOR Arbiter PUF modeling.

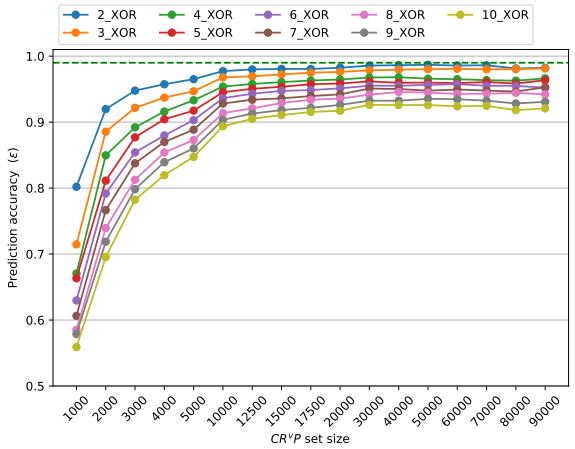

Fig. 4: Demonstrating the convergence of prediction accuracy ϵ of the XOR PUF variants with respect to increasing css.

Noting that the competing methods (R,T,M* and S) in Table. II are not optimized for enrollment. These methods are generally proposed to model the whole XOR Aribter PUF using a back-propagation technique to adjust the internal values θ directly according to the response behavior of the whole PUF model. Therefore they do not consider the internal responses.

From Table II, R is the modeling solution of [12] which uses Logistic Regression with RMSProp as the training function. In T [14] also, the modeling approach is the same as R, while the training is optimized to be faster. In M* [19] and S [15], Artificial Neural Networks are used as the underlying model and Adam optimizer as the training function.

As shown in Table. II, sub-space modeling indeed is capable of reducing the cost of training. Our solution (SS) seems to have the highest efficiency in terms of css, especially for modeling 5 6 and 7 XOR sizes. The exception is for 4-XOR where R shows to have a better result in terms of css. Given however that for the same XOR size, our solution takes significantly less training time T compared to R. The closer case in terms of css to our solution, is S. However in terms of T and model size ms, our solution shows better results overall. In terms of T, our model seems to have overall the best performance as well. Given also that closer case in terms of T to our model is M*. However for M*, the css seems to

be much more overall. It can thus be implied here that sub-space modeling overall can be considered to have the highest efficiency in terms of all factors that constitute the cost of modeling, for modeling strong PUF with increased complexity.

Next in sub-space modeling, we measure the prediction accuracy with respect to increasing the number of CR^vPs for a larger scale of XOR Arbiter PUF variants. Fig. 4 shows the evolution of prediction accuracy ϵ with respect to increasing CR^vP set size for XOR sizes $\{2, 3, 4, 5, 6, 7, 8, 9$ and $10\}$. We observe that the convergence point for ϵ degrades proportionally with increasing XOR size k. For instance, for 4-XOR variants the convergence point for ϵ is at 0.98 while for 10-XOR it is at 0.93. Another conclusion is that ϵ for all variants start to converge to its maximum value at around 10,000 CR^vP. The peak for ϵ could be seen in the range of $10,000$ to $40,000$ CR^vPs. While from $40,000$ CR^vPs above, the ϵ seems to degrade slightly. Which could be due to overfitting the model with too many CR^vPs. This means that sub-space modeling at this stage has a downside which is the degradation in the maximum prediction accuracy with increasing PUF complexity.

To further analyze prediction accuracy degradation, we measured the distribution of ϵ over all the sub-components of all the variants of k-XOR for k in $\{2, 3, 4, 5, 6, 7, 8, 9$ and $10\}$. Fig. 5 shows the histogram of all the sub-model's prediction accuracy with respect to several CR^vP set sizes. We observe that the prediction accuracy is less concentrated for smaller CR^vP set sizes like 1000 or 2000. This easily justifies the low prediction accuracy of the whole model. Since the rate of miss-prediction at these CR^vP set sizes are quite high, and as we know that the accuracy of the whole model can be defined as a product of the accuracy (see 3) of the sub-models. Therefore, the effect of internal miss-prediction on the whole model's prediction accuracy will be significant. For instance for a 2-XOR Arbiter PUF we have:

$$\epsilon_{WPUF} = \overbrace{(\epsilon_{SPUF_1} \times \epsilon_{SPUF_2})}^{\text{Probability of two correctly predicted responses at the same time}} + \underbrace{((1-\epsilon_{SPUF_1}) \times (1-\epsilon_{SPUF_2}))}_{\text{Probability of two incorrectly predicted responses at the same time}} \quad (3)$$

Where ϵ_{WPUF} refers to the prediction accuracy ϵ of the whole PUF model, and ϵ_{SPUF_i} refers to the prediction accuracy ϵ of the ith sub-model. Given also that this equation extends for larger XORs, where there needs to be computed the product of miss-prediction probability of every even number of sub-models. Here in (3), if we give the average prediction accuracy to the parameter ϵ_{SPUF_i}, it yields approximately the prediction accuracy value for the whole model as indicated in Fig. 4.

Looking at Fig. 5 (e), it is apparent that for the $40,000$ CR^vPs where the peak value of ϵ is achievable, the ϵ values are distributed around .992. This also justifies why the accuracy for modeling variants of k-XOR with increasing k, degrades as well. Again, according to formula (3), variation as small as 0.01% in the prediction accuracy of the sub-space

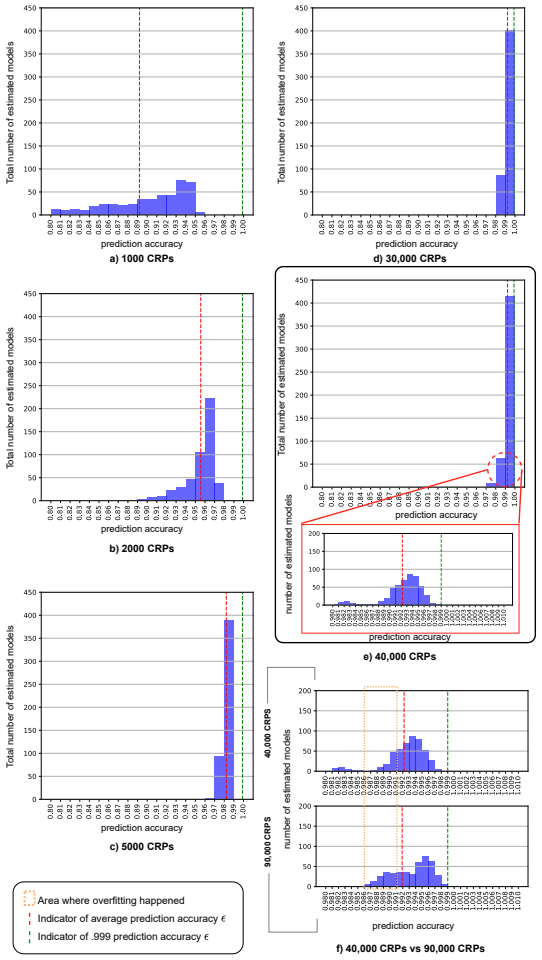

Fig. 5: Histograms showing the distribution of prediction accuracy ϵ for various training CRP set sizes.

models, can lead to variation of up to 1% of the whole model. For instance, according to and extended version of formula (3) for 10-XOR, we need in average, 99.9% prediction accuracy achieved for each sub-model to then achieve 99% prediction accuracy for the whole model. However, with our observation of the prediction accuracy of models trained with the most optimal training set size, the sub-space models' prediction accuracy are not concentrated on 99.9%.

For larger than $40,000$ CR^vPs, it appears that we overfitted a population of sub-models (see the outlined area in 5 (f)) which deteriorated the estimation of the whole model's prediction accuracy. Fig. 5 (f) shows that a considerable number of models trained with $90,000$ CR^vP, appear to have prediction accuracy lesser than the median 99.2% at $40,000$ CR^vPs. This means that only increasing the CR^vP set size is not a solution to achieve the highest accuracy for the whole model, as it can lead to overfitting the sub-models.

Our general observation on the sub-space modeling here infers that sub-space modeling seems to be a considerably resource efficient modeling technique. Given however, that its overall accuracy is more sensitive to minor prediction accuracy variations in the sub-models. Therefore special care should be

given to stabilize the prediction accuracy of the sub-model on the maximum achievable accuracy. For instance, if the maximum achievable accuracy is 99.9%, we set this as the target accuracy for all the sub-space models. This then requires that each sub-space model is trained with discrete attention to the number of CR^vPs in order to obtain exactly the target prediction accuracy for the whole PUF model.

Moreover, the number of the sub-components seem to affect the prediction accuracy of the whole model. It is observable that with larger number of sub-components, the accumulation of the probability of miss-prediction errors of the sub-models lead to higher values. Therefore, it is important to manage the number of sub-models. For future extensions of the work, it is potential to try with various dividing factors for sub-space modeling. One example is to divide a k-XOR Arbiter PUF with k being an even number, to $\frac{k}{2}$ of 2-XOR PUFs and model each 2-XOR Arbiter PUF separately.

V. Conclusion and future perspective

In this work we presented a technique for modeling strong PUF, using captured internal data of the sub-components of the PUF with high complexity. We showed that sub-space modeling requires significantly less data in order to yield an accurate estimated model of the PUF. For instance, we showed that it is possible to model 128-stage 10-XOR Arbiter PUF with 93% prediction accuracy using $40,000$ CR^vPs which is equivalent to 683KB of data. This proved that sub-space modeling is a potential cost-efficient solution for the designers community whose priority for enrolling strong PUF is to provide an estimation model of the PUF. However, we observed that the maximum achievable accuracy can be limited with sub-space modeling, due to the internal models' prediction accuracy variation. As expected, we observed that even a small prediction accuracy variation of around 0.1% in the internal models can cause the accuracy degradation of the whole model up to 1%. Depending on the complexity of the whole PUF also, this degradation can be magnified. And moreover, only adding more training data showed not to be the solution to overcome the prediction accuracy degradation. Therefore, this would be an open problem to solve in sub-space modeling of strong PUF with high complexity.

For future works, we consider other estimated model structure as well which can produce the same result as LR and have the capability to be extended. In specific, Artificial Neural Networks are the promising models which can replace LR to deal with the whole model prediction accuracy degradation. Moreover, we include other variants of strong PUF to generalize our modeling method for PUF enrollment.

References

[1] C. Herder, M.-D. Yu, F. Koushanfar, and S. Devadas, "Physical unclonable functions and applications: A tutorial," *Proceedings of the IEEE*, vol. 102, no. 8, pp. 1126–1141, 2014. [Online]. Available: http://ieeexplore.ieee.org/document/6823677/

[2] J. Delvaux, R. Peeters, D. Gu, and I. Verbauwhede, "A survey on lightweight entity authentication with strong pufs," *ACM Comput. Surv.*, vol. 48, no. 2, Oct. 2015. [Online]. Available: https://doi.org/10.1145/2818186

[3] M. S. Alkatheiri, Y. Zhuang, M. Korobkov, and A. R. Sangi, "An experimental study of the state-of-the-art pufs implemented on fpgas," in *2017 IEEE Conference on Dependable and Secure Computing*. IEEE, 2017, pp. 174–180. [Online]. Available: http://ieeexplore.ieee.org/document/8073844/

[4] R. Horstmeyer, B. Judkewitz, I. M. Vellekoop, S. Assawaworrarit, and C. Yang, "Physical key-protected one-time pad," *Scientific reports*, vol. 3, no. 1, pp. 1–6, 2013.

[5] B. T. Bosworth, I. A. Atakhodjaev, M. R. Kossey, B. C. Grubel, D. S. Vresilovic, J. R. Stroud, N. MacFarlane, J. Villalba, N. Dehak, A. B. Cooper, M. A. Foster, and A. C. Foster, "Unclonable photonic keys hardened against machine learning attacks," *APL Photonics*, vol. 5, no. 1, p. 010803, 2020. [Online]. Available: https://doi.org/10.1063/1.5100178

[6] M. Majzoobi, M. Rostami, F. Koushanfar, D. S. Wallach, and S. Devadas, "Slender puf protocol: A lightweight, robust, and secure authentication by substring matching," in *2012 IEEE Symposium on Security and Privacy Workshops*, 2012, pp. 33–44.

[7] A. Alipour, D. Hely, V. Beroulle, and G. Di Natale, "Power of prediction: Advantages of deep learning modeling as replacement for traditional PUF CRP enrollment," in *TrueDevice2020*, 2020. [Online]. Available: https://hal.archives-ouvertes.fr/hal-02954099

[8] M. S. E. Quadir and J. A. Chandy, "Embedded systems authentication and encryption using strong puf modeling," in *2020 IEEE International Conference on Consumer Electronics (ICCE)*, 2020, pp. 1–6.

[9] A. Alipour, V. Beroulle, B. Cambou, J. Danger, G. D. Natale, D. Hely, S. Guilley, and N. Karimi, "Puf enrollment and life cycle management: Solutions and perspectives for the test community," in *2020 IEEE European Test Symposium (ETS)*, 2020, pp. 1–10, ISSN: 1558-1780.

[10] J.-Q. Huang, M. Zhu, B. Liu, and W. Ge, "Deep learning modeling attack analysis for multiple fpga-based apuf protection structures," in *2018 14th IEEE International Conference on Solid-State and Integrated Circuit Technology (ICSICT)*. IEEE, 2018, pp. 1–3. [Online]. Available: https://ieeexplore.ieee.org/document/8565728/

[11] M. Khalafalla and C. Gebotys, "PUFs deep attacks: Enhanced modeling attacks using deep learning techniques to break the security of double arbiter PUFs," in *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2019, pp. 204–209. [Online]. Available: https://ieeexplore.ieee.org/document/8714862/

[12] U. Ruhrmair, F. Sehnke, J. S olter, G. Dror, S. Devadas, and J. u. Schmidhuber, "Modeling attacks on physical unclonable functions," in *Proceedings of the 17th ACM conference on Computer and communications security - CCS '10*. ACM Press, 2010, p. 237. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1866307.1866335

[13] F. Ganji, D. Forte, and J.-P. Seifert, "Pufmeter a property testing tool for assessing the robustness of physically unclonable functions to machine learning attacks," *IEEE Access*, vol. 7, pp. 122 513–122 521, 2019.

[14] J. Tobisch and G. T. Becker, "On the scaling of machine learning attacks on pufs with application to noise bifurcation," in *Radio Frequency Identification*, S. Mangard and P. Schaumont, Eds. Cham: Springer International Publishing, 2015, pp. 17–31.

[15] P. Santikellur and R. S. Chakraborty, "A computationally efficient tensor regression network-based modeling attack on xor arbiter puf and its variants," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 40, no. 6, pp. 1197–1206, 2020.

[16] K. T. Mursi, Y. Zhuang, M. S. Alkatheiri, and A. O. Aseeri, "Extensive examination of XOR arbiter PUFs as security primitives for resource-constrained IoT devices," in *2019 17th International Conference on Privacy, Security and Trust (PST)*. IEEE, 2019, pp. 1–9. [Online]. Available: https://ieeexplore.ieee.org/document/8949070/

[17] N. Wisiol, K. T. Mursi, J.-P. Seifert, and Y. Zhuang, "Neural-network-based modeling attacks on xor arbiter pufs revisited." *IACR Cryptol. ePrint Arch.*, vol. 2021, p. 555, 2021.

[18] B. Gassend, D. Clarke, M. van Dijk, and S. Devadas, "Silicon physical random functions," in *Proceedings of the 9th ACM conference on Computer and communications security*, ser. CCS '02. Association for Computing Machinery, 2002, pp. 148–160.

[19] K. T. Mursi, B. Thapaliya, Y. Zhuang, A. O. Aseeri, and M. S. Alkatheiri, "A fast deep learning method for security vulnerability study of XOR PUFs," *Multidisciplinary Digital Publishing Institute (MDPI) Electronics*, vol. 9, no. 10, p. 1715, 2020, number: 10 Publisher: Multidisciplinary Digital Publishing Institute. [Online]. Available: https://www.mdpi.com/2079-9292/9/10/1715

[20] http://www.pcp.in.tum.de/code/lr.zip.