Attacking Delay-based PUFs with Minimal Adversarial Knowledge

Fei Hongming, Owen Millwood, Prosanta Gope *Senior Member, IEEE*,
Jack Miskelly, Biplab Sikdar *Senior Member, IEEE*

***Abstract*—Physically Unclonable Functions (PUFs) provide a streamlined solution for lightweight device authentication. Delay-based Arbiter PUFs, with their ease of implementation and vast challenge space, have received significant attention; however, they are not immune to modelling attacks that exploit correlations between their inputs and outputs. Research is therefore polarized between developing modelling-resistant PUFs and devising machine learning attacks against them. This dichotomy often results in exaggerated concerns and overconfidence in PUF security, primarily because there lacks a universal tool to gauge a PUF's security. In many scenarios, attacks require additional information, such as PUF type or configuration parameters. Alarmingly, new PUFs are often branded 'secure' if they lack a specific attack model upon introduction. To impartially assess the security of delay-based PUFs, we present a generic framework featuring a Mixture-of-PUF-Experts (MoPE) structure for mounting attacks on various PUFs with minimal adversarial knowledge, which provides a way to compare their performance fairly and impartially. We demonstrate the capability of our model to attack different PUF types, including the *first* successful attack on Heterogeneous Feed-Forward PUFs using only a reasonable amount of challenges and responses. We propose an extension version of our model, a Multi-gate Mixture-of-PUF-Experts (MMoPE) structure, facilitating multi-task learning across diverse PUFs to recognise commonalities across PUF designs. This allows a streamlining of training periods for attacking multiple PUFs simultaneously. We conclude by showcasing the potent performance of MoPE and MMoPE across a spectrum of PUF types, employing simulated, real-world unbiased, and biased data sets for analysis.**

***Index Terms*—Physical Unclonable Function (PUF), Machine Learning-Modelling Attacks (ML-MA), Minimal Adversarial Knowledge.**

I. Introduction

Lightweight authentication is gaining momentum as a vital research area, primarily due to the ubiquitous integration of the Internet of Things (IoT) in our daily lives. Numerous methodologies have emerged to bolster its resilience. Traditionally, secret keys stored in non-volatile memory are employed to encrypt sensitive data, and cryptographic techniques, such as asymmetric cryptography, have been used for device authentication [1]. However, cryptographic implementations can be resource-intensive, particularly given the nature of IoT devices, which are often resource-constrained. Even with cryptography, devices remain susceptible to various threats, including invasive attacks [2]. In contrast, Physical Unclonable Functions (PUFs) offer a streamlined approach to security, suitable for both authentication and secure key generation. Their efficacy in authentication revolves around optimal energy consumption, computational power, and robust defence against threats. This is largely attributed to PUFs deriving volatile secrets from a device's inherent physical characteristics rather than relying on stored secrets in non-volatile memory. These characteristics, resulting from random variations during integrated circuit (IC) manufacturing, ensure that no two ICs are identical [1]. Moreover, the unique delay sequences in transistors and wires of each IC make PUFs capable of generating unpredictable sequences, offering a formidable defence against malicious attacks. Furthermore, PUFs are efficient, negating the need for intricate cryptographic operations.

PUFs utilize sequences of binary numbers as input and output, referred to as challenge-response pairs (CRPs) [3]. Based on the number of CRPs they can produce, PUFs are classified as 'Weak' and 'Strong'. This terminology has no bearing on the security properties of the PUF but rather indicates the total number of supported unique CRPs. Weak PUFs are limited to generating a linear number of CRPs, making them apt for key generation and storage. Strong PUFs can produce an exponentially growing number of unique CRPs (based on PUF size), making them ideal for creating one-time authentication tokens. However, a key vulnerability persists for PUFs, particularly for delay-based Arbiter PUFs, known as 'machine learning modelling attacks' (ML-MA). In such attacks, adversaries collect CRPs produced by PUFs and employ machine learning techniques to deduce the challenge-response correlation [4]. As such a model can predict the response to future challenges, it can allow an adversary to pose as the PUF-authenticated device. The high prediction accuracy of this technique jeopardizes the security of PUFs. Consequently, diverse PUF variations have been proposed to fortify their defences. For instance, XOR Arbiter PUFs (XOR-APUFs) integrate the responses of multiple Arbiter PUFs (APUFs) [5][6], while (XOR) Feed-Forward Arbiter PUFs (FF-APUFs) incorporate a "feed-forward loop" (FF-loop) concept and Interpose PUFs (iPUFs) utilize the upper-lower-layers design. All these designs are geared towards enhancing PUF non-linearity [4]. Nevertheless, so long as the training CRP set is large enough, accurate predictions of these complex PUFs are still feasible. Without more extreme countermeasures, they only raise the modelling effort and do not fundamentally prevent it.

H. Fei and B. Sikdar are with the Department of Electrical and Computer Engineering, National University of Singapore, Singapore.

P. Gope and O. Millwood are with the Department of Computer Science, The University of Sheffield, Regent Court, Sheffield S1 4DP, United Kingdom.

J. Miskelly is with the Centre for Secure Information Technologies, Queen's University Belfast, United Kingdom.

A. Related Work

In general, the crux of modelling attacks on PUFs is identifying the relationship between challenges and responses. This implies that the attacker needs access to the CRPs in the PUF. Multiple studies have explored how to conduct modelling attacks on PUFs using ML methods. Rührmair et al. first employed Logistic Regression (LR) to model XOR-PUFs [4]. They began by formalizing the mathematical model of the arbiter PUF and then captured the XOR logic for the machine learning model. For instance, they utilized multiplication operations of $1, -1$ to symbolize XOR, achieving high modelling accuracy. However, their mathematical model is less effective when faced with noisy data and intricate PUFs, such as those with many XOR-APUF stages. In [7], both LR and Evolution Strategies (ES) were deployed to target various PUFs (some of which are examined in this paper) using both simulated and silicon data. Their outcomes reaffirmed the viability of ML-MA. PUFs' reliability also emerged as a vulnerability, revealing differences in delay paths. Becker [8] introduced reliability-based machine learning attacks, demonstrating a link between response reliability and delay differences, suggesting that an unstable response indicates a minuscule delay difference. In [9], researchers presented a comprehensive framework to conduct logical approximation modelling attacks on several delay-based PUFs. The framework leveraged a logical approximation and global approximation rooted in artificial neural networks (ANN). The former estimated basic logical operations like AND and OR in circuits, applied linear functions, and built an ANN model of the logical architecture. For more intricate logics, the global approximation discerned an appropriate continuous function to map the challenge-response relation, subsequently formulating an ANN structure to emulate the chosen function. The primary goal of [9] was to depict the nonlinear components in PUF designs through soft models to expedite modelling. In [5], the authors delved into another type of modelling attack on Arbiter PUF compositions. This attack was founded on a deep feed-forward neural network and did not rely on the mathematical model or structural knowledge of PUFs as seen in prior work [4], [7], [8]. Distinct layers with varying neurons were meticulously designed to enable a successful attack. Aseeri et al. [10] considered a different neural network approach to model XOR PUFs. This structure is more standardized, expanding with the increase of component arbiter PUFs. Mursi et al. in [11] refined the structure and various ML hyper-parameters, outperforming [5], [10]. Nonetheless, the efficiency of ML-MA methods varies across devices and datasets. In [12], a new strategy is proposed, using "transition theory" to represent the mathematical model of the PUFs to an equivalent representation of XOR PUFs, which can connect the modelling resistance across different PUFs and further enable an "out-of-date" model to attack a new PUF.

In [13], Wisiol et al. provided unbiased comparisons of recent ML-MAs, underscoring the promise of neural network-based attacks on PUFs. They also unveiled a new model for XOR FF-APUFs but admitted their inability to breach heterogeneous XOR Feed-Forward Arbiter PUFs, even under moderate parameter settings. A recurring observation across these studies was the sensitivity of the methods to model settings. Even minor alterations could result in attack failures, indicating designs tailored for specific PUFs. In [14], a new evolutionary search (ES) based modelling method $\mathrm{CalyPSO}$ was proposed. $\mathrm{CalyPSO}$ targets delay-based PUFs and was able to mount successful modelling attacks on high-order XOR-PUFs, interpose-PUFs and LR-PUFs. The method requires knowledge of the architectural topologies, and the training cost is exceptionally high relative to other approaches. Significantly, compared to this work, they attempt a "downgrade attack" where they attack $n-XOR$ PUFs using a model tuned for $(n+1)-XOR$ and achieve moderate success, making this method at least partially generic. While the concepts in this paper are interesting, we have some reservations about the results claimed due to errors found in the codebase associated with the paper and the resulting difficulty in replicating the results, e.g., overlaps between the training dataset and test dataset, miscalculation of the amount of training data, etc.

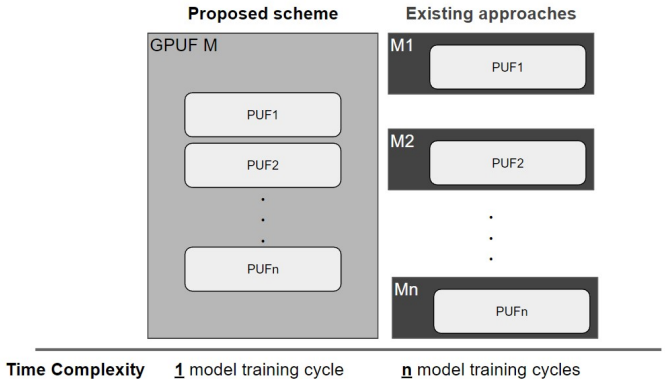


Fig. 1. Generic framework vs. multiple single models for PUF modelling

Apart from CRPs, side channel information is also used for modelling PUFs. In [15], an auxiliary learning framework ALScA is proposed, which is a multiple-task learning model utilizing both CRPs and side channel information to help modelling tasks. ALScA has a shared-bottom structure and focuses on the correlations between the mathematical model and side-channel models and achieves good accuracy improvement on Arbiter-based PUFs.

B. Problem Statement and Motivation

Strong PUF-based authentication threat models often assume an attacker capable of gathering large numbers of CRPs and a single target PUF of fixed type. Further, the structure of the PUF is known to the adversary. In many ways, this is a worst-case scenario for remote attacks: a logical threat model for most PUF papers where the aim is either to prove the security of a given design or to disprove such claims previously made. In reality, it is unlikely that most attackers will have the unlimited ability to read network traffic or have such detailed knowledge of hardware implementation. If the case is that only one PUF exists in the network. PUF authentication protocols are generally agnostic of the specific

hardware implementation. So, there is no reason to assume that every device has the same PUF or PUFs in a mixed device network. This cannot be easily inferred from network traffic alone, nor is it something that manufacturers are likely to disclose. Thus, when attacking PUFs, we should treat them as an ideal black box; only input and output data are available and no other information. Besides, regardless of whether a PUF is provably secure, in relative terms, what is the real risk from an opportunistic attacker? Such an attacker is simply probing for weaknesses, who does not have unlimited ability to avoid detection, who is not a hardware security expert but has enough technical skill to use any tools that other experts have produced. Does such an adversary pose a real threat with the available tools? If not, could a tool be built to enable that threat? We should expect the adversary to prefer to attack without the need to know much more details about PUFs other than CRPs. As discussed above, this kind of extra information can be hard to obtain in practical scenarios.

Prevailing ML-MA techniques necessitate information beyond just CRPs. We classify this information into 'Explicit' and 'Implicit' categories. Explicit information encompasses setup parameters of PUFs, such as the placement of Feed-Forward Loops. On the other hand, Implicit information refers to type-specific details, like the count of APUFs in XOR PUFs or the number of Feed-Forward Loops in FF-PUFs. Some methodologies leverage 'Explicit' knowledge to construct precise models aligning with the PUFs. A majority employ 'Implicit' information for model creation, asserting that any alterations to their model would render the attack ineffective. Consequently, these models often falter in real-world applications. One might argue that iterating through all potential models is an alternative, but even with a comprehensive catalog of PUF designs and specific attacks, the uncertainty of which PUF to initially target makes this approach inefficient. It either results in trial and error or necessitates a deep learning approach demanding considerably more training data. As previously stated, in realistic settings, discerning the exact type of PUF a device employs, even with a uniform authentication protocol, presents a challenge for adversaries. For example, the same 64-stage 6 XOR-APUF in [5] was reported to need 20 minutes to train an attack model. Of course, the difficulty depends on available computing resources, but repeatedly training and testing models until the right one is found is costly. The alternative is an entirely black-box deep learning attack, which infers both the PUF model and the values of a given PUF. The core motivation of this work is to explore the lower bound of viable attackers. To this end, we examine techniques for training PUF prediction models in parallel and without prior design knowledge (as shown in Fig. 1). This is an example of a single expert tool that a low-capability attacker could use to probe for modelable PUFs. We also look at the lower bound of CRPs needed to achieve a high chance of collision between actual and predicted CRPs. The aim is to provide insight into the real risk level for PUF-based authenticated devices against opportunistic, low-effort attackers. Beyond the scope of practical adversary situations, we have observed a research schism. The community is divided between crafting modelling-resistant PUFs and designing machine learning attacks against them, sparking debates over their security robustness. This divide often inflates concerns and fosters undue confidence in PUF security, primarily due to the absence of a standardized tool to evaluate a PUF's security. In discussions surrounding newly introduced PUFs, disputes arise over their relative security. While unique attack strategies may exist for each, they might employ varying techniques, such as neural networks or evolution strategies. Finding an equitable comparison metric is thus challenging. We aim to introduce a universal framework capable of modelling any delay-based PUF without necessitating setting alterations. In this manner, the requisite number of CRPs can serve as a definitive standard for assessing security levels, and thus, the quantitative index of modelling resistance of a PUF design. On the other hand, managing multiple models for multiple PUF-embedded devices could be challenging, where an adversary needs to train, store and load individual PUF models for a target device. Thus, to address this issue, in this research, we aim to build a generic PUF model where an attacker needs to maintain only a single model for all PUFs. To summarize, our motivation for this article can be specified through the following research questions (RQs):

1) **RQ 1:** *Can we perform a modelling attack on a PUF without knowing its architectural topology information?*
2) **RQ 2:** *Is it possible for an attacker to maintain* ***a single model*** *to attack multiple cross-architectural PUFs? And would no update of the constructed single model even be required to perform the attack on other PUFs?*
3) **RQ 3:** *Considering the practical concerns of training efficiency, time cost, and model loading problem, how realistic is it to attack multiple PUFs together with one single model?*

C. Contributions

Existing literature posits two primary ML attack methods against PUFs: mathematical-model-based and deep-learning-based attacks. Attacking using a predetermined mathematical model is highly efficient. However, it requires an attacker to know the type of PUF and details of its specific implementation (e.g., the number of stages in a delay PUF, the number of XORs in an XOR PUF, etc.). On the other hand, PUF inference using deep learning can attack any PUF without knowing as much detail as required by mathematical-model-based methods, but requires large sets of training data and some high-level implementation details. In this work, we propose a generic model which does not require any information about the targeted PUF except the CRPs. Further, we expand the model for multi-task learning that could be applied to PUF attacks to exploit commonalities in structure and logic across PUF instances, thereby creating an attack which is generic (like deep learning inference), with reasonable training data and computation costs (like using a predefined model), minimal expertise required to use, and which retains high prediction accuracy. The specific contributions are:

1) We define a pragmatic PUF attack scenario based around determining the **"Minimum Viable Adversary"**. That is the bare minimum of skill, knowledge, and resources

required to compromise the authentication system under test.

2) We propose a **generic** framework based on multi-experts collaborative learning for modelling delay-based PUFs with minimal knowledge and resources. This represents the kind of tool that significantly lowers the bar for what constitutes a Minimum Viable Adversary.
3) We provide a replication of several proposed ML attack methods found in the existing literature [4], [10], [11] and compare their performance with respect to our proposed solution using both simulated and in-silicon PUF data in order to validate our claim that attacks using predefined mathematical models are not generic.
4) We demonstrate a successful attack using our tool on the heterogeneous XOR Feed-Forward Arbiter PUF. To the best of our knowledge, this is the *first* published attack against this design.
5) We conduct cross-architectural experiments and show that our model can achieve high accuracy among different kinds of delay-based PUFs without any modifications to the model. For instance, we attack five distinct delay-based PUFs, four unique 4-XOR-APUFs, five different FF-XOR-APUFs and Interpose PUFs, with all attacks attaining a prediction accuracy over 90%.
6) We successfully attack multiple PUFs together in one single model, where the PUFs have different settings and structures. We conduct detailed experiments and achieve accuracy beyond 90% in all the experiments.
7) We provide all code, FPGA implementations of PUFs, and datasets used in this paper for the use of the research community[1]. We hope this tool will be useful for those working on variant delay-based PUFs as an "out of the box" attack to test against, without needing a lot of modification or computational resources to run.

II. Preliminaries

This section lays the foundation for our discussion by providing an overview of Arbiter PUF and its various components. We delve into the intricate world of modelling attacks aimed at PUFs, highlighting representative methods that challenge their security. Furthermore, we introduce the cutting-edge Multi-gate Mixture-of-Experts Model (MMoE) as a promising solution. Lastly, we define the adversary model as the basis for our analysis.

A. Delay-based PUFs

As described in Section I, PUFs can be classified into two distinct categories: Strong PUFs and Weak PUFs. This classification does not reflect their security levels but rather hinges on the number of CRPs they support. The exponential number of CRPs in Strong PUFs renders them the preferred choice for device authentication. Their inherent property of random deviations during hardware manufacturing grants them an unclonable physical structure, making it challenging for attackers to model their behaviour. However, they remain susceptible to modelling attacks facilitated by ML techniques. In these attacks, an adversary collects a significant number of CRPs and constructs a mathematical model, allowing them to predict the patterns of CRPs accurately. This section aims to elucidate the concepts behind various types of Strong PUFs. An example of an implementation of a strong arbiter PUF is shown in Figure 2. This design exploits the manufacturing variability in the gate delays for randomness and establishes a race condition in a symmetric circuit. The circuit splits an input edge to two multiplexers and creates two identical paths to the output latch based on the input challenge bits $c[0], \cdots, c[n]$. Though the two paths and their propagation times are identical as designed, the random manufacturing variability in the gate delays at the multiplexers will result in one edge arriving first at the latch, with the latch acting as the "arbiter". The figure shows one output (response) bit that depends on the n challenge bits, and multiple such circuits may be used in parallel to obtain additional response bits.

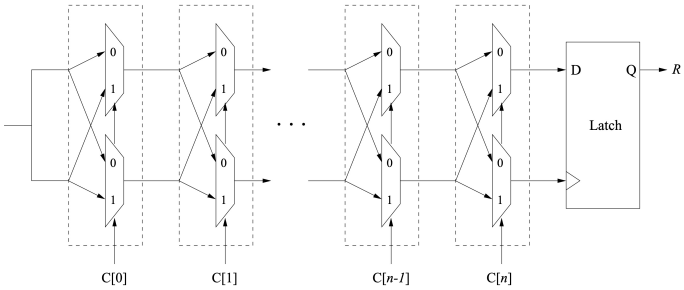


Fig. 2. Arbiter PUF delay chain diagram.

B. Classical Machine-Learning Modeling Attacks on PUF

ML-MA on PUFs have been a significant pain point when developing Strong PUFs, almost since their conception. ML-MA are carried out by adversaries collecting a subset of CRPs from an individual PUF's total CRP space to use as an input to a sophisticated ML algorithm, such that a mathematical model can be generated which learns correlative properties between different challenges and responses. Over the years, almost all Strong PUFs proposed are vulnerable to ML-MA using many different types of ML algorithms, ranging from traditional ML, which is specifically tailored to a given PUF design, to deep learning methods. The first significant work exposing the vulnerability of PUFs to ML-MA was demonstrated by Rührmair et al. in [4], where the Logistic Regression and Covariance-Matrix Evolutionary Strategy (CMA-ES) algorithms were exploited to model Arbiter PUFs, Ring Oscillator PUFs, XOR Arbiter PUFs, Lightweight Secure PUFs and Feed-Forward Arbiter PUFs. These attacks required varying numbers of CRPs for training the models, with the simple Arbiter PUFs, at a minimum, requiring simply 640 CRPs to break the PUF. The more obfuscated PUFs (XOR-Arbiter PUF and Feed-Forward Arbiter PUF), however, generally required many more CRPs before model convergence occurred at up to 500,000 in most cases. While less efficient than traditional

[1]**The code and datasets used in this paper are provided in full for the use of the research community at: https://github.com/AnonymousAppdx/Generic-Framework-for-Modelling-PUFs**

ML-MA (on PUFs), deep learning-based modelling attacks can learn latent representation without requiring knowledge of the underlying PUF structure, broadening their use cases [16], [17]. More extensively obfuscated APUF designs have shown improved defences against traditional ML attacks; however, feed-forward neural networks (FNNs) have been shown to successfully model up to 5-XOR APUFs, and (4,4)-iPUFs [16].

C. Multi-task Learning

Multi-task learning has been considered to be an important research topic in the machine learning community for a long time. By transferring the common knowledge shared between different but related tasks, multi-task learning is expected to improve efficiency and model quality on each task. Deep learning has continuously made breakthroughs in various tasks and applications in recent years. Consequently, multi-task learning methods based on deep network architectures have gradually become the research mainstream. The shared bottom network, proposed by Caruana [18], is one of the most widely adopted multitask learning methods. It comprises of a shared bottom model structure, where the hidden bottom layers are shared between tasks and several tower networks specific to the task. The shared-bottom model structure enables knowledge transfer among tasks and dramatically reduces the risk of overfitting. Unfortunately, it may suffer from negative transfer as all tasks must utilize the same shared bottom layers, and the differences between tasks are artificially obliterated.

D. Mixture-of-Experts and Multi-gate Mixture-of-Experts Model

Mixture-of-Experts (MoE) layer was first proposed by Robert et al. [19] as an associative version of competitive learning. By dividing the main tasks into several appropriate subtasks, each of the subtasks can be solved using simple expert networks. The spirit of dividing and learning is suitable for PUF modelling since the basic APUF component can be easily learned using a network with only dozens of neurons. The core of MoE is the gate function, which can be trained to 'select' the most suitable experts for the main task. This kind of gate function is usually composed of only several neurons and does not cost much resources. It has normalized outputs which are activated by a $Softmax$ function, referring to the weights assigned to each expert. Then, each expert contributes to the final output according to the weights.

The Multi-gate Mixture-of-Experts Model is proposed by Wang et al. [20] to solve the negative transferring produced in multi-task modelling. Additionally, it can also capture the similarities across tasks and improve general performance. MMoE adds more gate functions based on the original MoE Model [19]; the structure is shown in Figure 3. The core idea of multiple experts is to set up a MoE layer as an ensemble method of multiple individual models. These models are viewed as experts with specific capabilities to solve different tasks. Then, a separate gating network g^t for each task t is used to select suitable experts. The output of task t is:

$$y_k = h^t\left(f^t(x)\right), \tag{1}$$

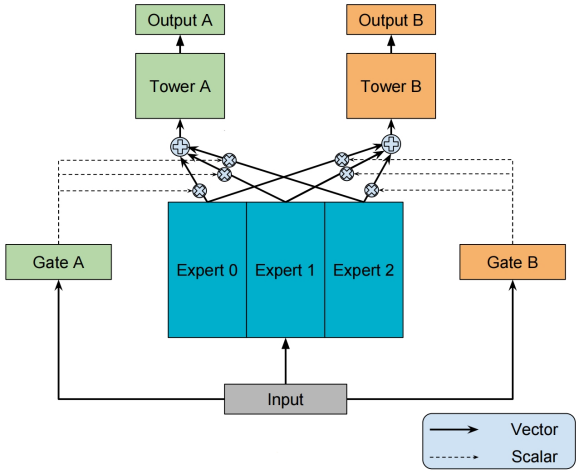


Fig. 3. Multi-gate MoE Model.[20]

where

$$f^t(x) = \sum_{i=1}^{n} g^t(x)_i f_i(x). \tag{2}$$

The gating networks are linear transformations of the input with a softmax layer:

$$g^t(x) = \text{softmax}\left(W_{g_t} x\right). \tag{3}$$

E. The Basic Threat Model

The threat models of machine learning attacks (MLAs) have often not been thoroughly examined or analyzed. This oversight allows adversaries to easily gather excessive information, resulting in an overestimation of the adversary's capabilities and a corresponding underestimation of security robustness. Here, we give a formal basic threat model for modelling strong PUFs.

Basic Threat Model for Strong PUFs

1) A network of devices exists.
2) This network comprises of nodes (to be verified) and verifiers.
3) Nodes are low-resource devices in a low-security environment, and verifiers are high-resource devices in a high(er) security environment.
4) All nodes possess a PUF. The PUF can be issued challenges and returns a response.
5) Responses to a given challenge are unique for a given node.
6) A verifier knows the expected response to any challenge for all nodes it is assigned to verify.
7) A node is considered authentic (trusted) if it can return the expected response to challenges issued by the corresponding verifier.
8) The adversary cannot directly access device contents for either nodes or verifiers.
9) The adversary does not have physical access to devices.

10) The adversary cannot alter the behaviour of devices but can trigger any behaviour that is exposed to the network.
11) The network is transparent to the adversary. The adversary acquires any data transmitted by devices on the network.
12) If the adversary can predict expected responses, they can impersonate a node and gain trusted access to the network.

The crux of the modelling attack problem for PUFs are points 11 and 12 of the basic model described above. In early works, it was assumed that simply discarding a CRP after use was enough to prevent response prediction. A large body of subsequent work has shown that, in fact, an ML model trained on a subset of CRPs can infer the response to any arbitrary challenge. Proposals to mitigate this can be broadly placed into the following categories:

Time Bounded: Rate limit PUF query such that a response can be generated only once every T seconds. In theory, so long as T is sufficiently large, it becomes impractical for an adversary to acquire a training set for any given PUF. Even if some attack is used to challenge the node rapidly, it will only respond at a fixed rate. This opens the risk of denial of service attacks against the authentication system. The designer must balance network performance against the lower bound on the number of CRPs needed for an attack.

Mathematically Unmodelable PUF: The standard threat model is safe so long as the PUF cannot be predicted without knowing the full CRP set in advance. Ideally, the mathematical model of the PUF should be able to prove this infeasibility. Many "ML resistant" PUF proposals aim for this objective, but to date any time such claims have been made, some new attack method has proven them false for at least some scenarios. In this case, a powerful model suitable for different PUFs can be a good benchmark for researchers to evaluate their proposed PUFs in the first place.

Complexity Bounded: This approach relies on increasing the number of physical variables and the complexity of their relationship in the PUF design. The higher the complexity, the larger the time to model, even with many CRPs. Modelling is theoretically possible and feasible, given enough attacker resources. However, this difficulty is known and can be scaled by the designer at the cost of additional resource usage. However, complex designs bring unreliability to the responses of the PUF, which may weaken its security against reliability-based ML-MA. The designer must balance resource usage vs. time-to-model vs. response reliability vs. the anticipated attacker resources.

Obfuscated: This approach relies on decoupling responses from physical information in the PUF design itself. This can be done reliably by encrypting the authentication such that it, in effect, becomes opaque to the adversary. However, in many ways this negates the key benefits of a PUF: their low resource, lightweight nature. There is a body of published work on lightweight obfuscation techniques but these produce mixed results in practice.

Reconfigurable: This approach uses a PUF structure that can be reconfigured to change the challenge-response mapping. The idea is that even if an adversary can model a PUF once, risk can be mitigated if the system has a mechanism to invalidate that model. In some proposals, this is applied periodically or as part of the challenge, while in others, it is an active countermeasure that is applied if an intrusion or false authentication is suspected.

F. How Realistic Are Classic Threat Models?

For Strong PUFs, the number of real-world examples of deployment is highly limited. This means there is no way to be sure that they hold up in practice at the time of writing. This is due in no small part to the state of the art in attacks outpacing countermeasures, making Strong PUFs seem a dubious prospect at best to designers. Something we need to ask, however, is whether the fact that a given PUF *can* be modelled means it is *likely to be* modelled if deployed. Our standard threat model is unsuitable for this evaluation because it assumes an adversary with deep knowledge of PUFs attacking a single known PUF, with no barriers in play except any countermeasure built into the PUF itself. This certainly allows for the evaluation of the risk by a highly capable adversary but does not provide much insight into what guarantees can be made. There is a significant gap between "not resistant to the most capable adversaries" and "not resistant to any adversaries at all". Here, we list some of the assumptions of the basic threat model that may not align with practical implementation and discuss those aspects.

The network is transparent to the adversary. Is it realistic to assume that an adversary will have unrestricted ability to read network traffic with no time limit? Network intrusion detection is a well-explored area. Even in the context of likely devices for PUF use (lightweight embedded systems), there are proposals for detecting snooping [21].

The adversary has expert knowledge of PUFs and Machine Learning modelling. While modelling attacks on PUFs are often trivial in terms of computations required, this is in the context of academic literature. The level of expert knowledge needed to know how to implement such an attack is non-trivial. No tools currently exist to enable non-expert attacks against PUF systems. That said, there is no fundamental technical barrier to such a tool. The motivation to create one is simply lacking for the time being.

The adversary already knows what type of PUF is in play. Many of the more efficient attacks rely on prior knowledge of a mathematical model for the PUF structure. While for some PUFs, these models are public knowledge, for others, they have never (or at least not yet) been published. Deriving one requires a high level of expert knowledge. Conversely, black box attacks inferring the structure are relatively inefficient, needing more training data to achieve high prediction accuracy. Even assuming there is a public model for the PUF being used, there is the question of how the adversary knows which PUF is in the device before launching the attack. If that information has not been disclosed, they are limited to black-box attacks or must first use reverse engineering to determine the PUF.

There is only one type of PUF in play. Most published works propose a single PUF structure or use a single structure as an example of a PUF in a protocol. There is no reason why two or more entirely different PUFs cannot be used on a given device. Some proposed structures (such as the CT-PUF proposed in [22]) can even allow one circuit to act as different PUFs at will. This particular scenario actually provides one of the key benefits to an attacker when utilising the scheme proposed in this work. In scenarios where different PUF types are exploited to increase the difficulty for an attacker, a generic framework that does not require PUF knowledge is particularly powerful (this is experimentally verified and discussed in Section V).

The adversary can cleanly identify target PUF CRPs from normal network traffic and other PUF CRPs. To collect a training set of CRPs, the adversary needs to be able to spot when a PUF authentication is happening and correctly identify the device being authenticated. Even a few errors here will result in poisoned models that perform significantly worse than expected [23].

Everything that looks like a CRP is derived from a PUF. The corollary to how simple it is to model a PUF is that it is equally simple to poison such a model through active defence. A properly configured device can easily transmit network traffic, which appears to be a PUF protocol packet but which contains random data in place of a PUF response. As the adversary has not yet trained a predictive model for any PUF in the system, they have no way to tell these fake responses from real ones, leading to poisoning of the training data.

G. Minimum Viable Adversary Model

In many instances, the threat models of MLAs have not been thoroughly examined or analyzed. This oversight allows adversaries to easily gather excessive information, resulting in an overestimation of the adversary's capabilities and a corresponding underestimation of security robustness. The excessive information includes the type information of PUFs, e.g. XOR-APUFs or FF-APUFs, and the configuration details, such as XOR-APUFs or FF-APUFs, alongside intricate configuration details like the number of XOR gates or the positioning of Feed-Forward Loops. The methodologies for acquiring such information—and the feasibility of doing so—beyond merely Challenge-Response Pairs (CRPs) warrant careful consideration. Thus the key motivation in this work is to identify the level of actual capability and knowledge an attacker needs in order to pose a threat to a PUF system in real life. In practice, the system designer knows (or at least can know) about PUF vulnerabilities and countermeasures just as much as the adversary. To model a realistic scenario, therefore, we need to incorporate these countermeasures in a way that is still general enough to apply generically to PUF systems. A detailed discussion of various factors we considered can be found in the Appendix. This led us to the following threat model for determining whether a given adversary can perform a viable attack against a realistic PUF network, which uses the following assumptions:

Viable Adversary PUF Threat Model

1) A network of devices exists.
2) This network comprises of nodes (to be verified) and verifiers.
3) Nodes are low-resource devices in a low-security environment, and verifiers are high-resource devices in a high(er) security environment.
4) All nodes contain a PUF. The PUF can be issued challenges and returns a response.
5) The PUF has been designed with possible attacks in mind and is as complex as possible within the resource constraints of the system.
6) There is a rate limit, $Rate_{time}$, which dictates how often a PUF may be challenged.
7) Responses to a given challenge are unique for a given node.
8) A verifier knows the expected response to any challenge, for any possible configuration, for all nodes it is assigned to verify.
9) A node is considered authentic (trusted) if it can return the expected response to challenges issued by the corresponding verifier.
10) The adversary does not have physical access to devices or access to device contents.
11) The adversary cannot tell which PUF design is being used on any given node but knows which structures are most common in PUF design.
12) The adversary cannot alter the behaviour of devices. They can trigger any behaviour that is exposed to the network, but doing so risks detection.
13) There is a time-to-detection after which the network becomes aware of an active adversary and can trigger reconfigurations that reset any modelling progress.
14) The adversary has a limited window of time to gather CRPs. The number of CRPs which can be gained is a function of the rate limit, *T*, and the worst-case time-to-detection.
15) If the adversary can predict expected responses under these conditions, they are a ***Viable Adversary*** and can impersonate a node to gain trusted access.

Of particular interest in this work is the ***Minimum Viable Adversary***, the lowest effort and capability adversary who still has a good chance of success in this threat model. Modelling this kind of adversary gives us insight into the true lower bound of security for a given PUF design. It is of course important to know whether or not a security mechanism is theoretically secure, but even when the answer is "no", it may still provide a useful deterrent. A system which can be broken by 45% of adversaries can be justifiably called insecure but can equally be viewed as a successful deterrent for 55% of threats. A security measure's practical value is always relative to how much it costs and how capable the adversaries are. To calculate the actual practical value of adding a PUF to a given system, we need to know how the relative cost of the PUF compares to other lightweight primitives, the properties of the Minimum Viable Adversary, and whether they fall above

or below the anticipated capability of real attackers. This is a more complex and situational question than the binary theoretically secure/theoretically insecure dichotomy, but one worth investigating.

H. What Constitutes a Successful Attack?

A PUF can be considered 'broken' if an adversary can successfully predict complete (bit-perfect) responses with a significant advantage above 50% accuracy. i.e., random guessing. Intuitively, as the per-bit prediction accuracy increases, the number of guesses needed for a collision (all bits guessed correctly) reduces. While there is no significant consensus (as of the publication of this work) on a precise per-bit accuracy value at which a PUF is considered vulnerable, and a threshold value of 70% per-bit is commonly applied. The soundness of this is debatable; in the realm of cryptography, even advantages of 50% +/- 2% are considered to be insufficiently secure for some applications. In the case of PUFs, it is something that must be considered in the context of an applied attack. In a scenario where the adversary can perform queries at high speed, the adversary can attempt a brute force prediction analogous to guessing an encryption key. In this case, the assumption that $>$50% prediction accuracy is a serious flaw is a justified one. If, however, we look at a scenario where the attacker is remote and has to interact with the PUF over a network and via a protocol they cannot modify (the standard assumption for most PUF attack papers) then it becomes trivially easy to deny them that capability. If there is any sort of intrusion detection or time-bounding in play, then the brute force approach becomes much harder to pull off.

For example, a 70% per-bit prediction accuracy only has a 1.49×10^{-20} chance of a full collision on any given attempt for a 128-bit ID. Something as simple as a 1 nanosecond delay enforced between authentication attempts takes the average time to collide into the order of hundreds of years. Further, while faking authentication once is all well and good, ideally the adversary needs to be able to repeatedly do so without being detected. In order to fake a PUF with a good chance of success in a network that has even the most basic countermeasures, the prediction rate has to be fairly close to the error rate of the actual PUF transmissions. That is, the PUF after on-device error correction has been applied. We suggest that this must be *at minimum* $>$ 80% for a 64-bit ID (giving better than 1 in 1 million chance of a full correct guess on any given attempt), and by the same reasoning, 90% for 128 bits, 95% for 256 bits, 98% for 512 bits, and so on. Within a practical amount of time, less than 80% accuracy is only viable as an attack if the PUF ID is in the order of 32 bits or less.

Due to this, in the remainder of this work, we assume a success threshold of 90% as a reasonable minimum. If a tool could be made which achieves this threshold consistently for any target PUF with an achievable amount of CRPs, then the Minimal Viable Adversary presents a quite serious threat. Thus, we define a successful attack with minimal knowledge as follows.

Definition 1 (Successful attack against PUFs)**.** *An adversary $\mathcal{A}$ can successfully attack a PUF $\mathcal{P}$ with $M_{\mathcal{A}}^{\mathcal{P}}$ CRPs, if and only if for any instance of PUF $\mathcal{P}$, $\mathcal{A}$ can achieve prediction accuracy beyond 90% on unseen CRPs with any $M_{\mathcal{A}}^{\mathcal{P}}$ number of CRPs.*

We believe that such a tool is possible and such a definition is reasonable, and the following sections will present an example of one based on the Mixture-of-Experts model. Something to emphasise in our general methodology is the goal of consistency: aiming purely for minimal CRPs can result in an attack tool which fails for outlier devices. As such, throughout the following sections, we always use the approach of finding the lowest number of CRPs needed to achieve the target accuracy, testing for a very large numbers of PUFs, and if a failure is detected, increase the CRPs incrementally until we find no failures. This hopefully captures the limitations of an attack tool intended for use by non-experts.

III. Proposed Generic Framework

As discussed in Section I-A, numerous modelling attack methods [4], [11], [10], [24] have demonstrated commendable prediction accuracy in PUF responses. A shared characteristic among these methods is their reliance on the type or structure of the target PUF as foundational information for modelling. Typically, traditional machine-learning-based models are intricately tailored for a specific PUF type, exhibiting prowess in predicting unacquired CRPs with remarkable precision. Beyond the issue of generality, as delineated in the adversary model in Section II-G, acquiring such information covertly within a network is impractical. Expecting an adversary to test every possible structure while targeting a PUF is also illogical, given the varying amounts of training data required for different PUFs to orchestrate a successful attack. Considering the 'Minimal Viable Adversary', we introduce a generic framework to address these challenges of generality and absence of PUF information in PUF modelling. Here, a singular neural network structure is employed to model diverse PUF types, irrespective of prior PUF type knowledge. An extended version is also proposed to perform multiple-PUF modelling on the combinations of several of the same or different PUFs. As discussed in Section. I-A, many types of delay-based-Strong PUFs share structural similarities. Therefore, it is intuitive to utilise this structural similarity and capture the relationship across these differing PUF instances during modelling when preparing an ML-MA.

A. Generic Framework for a Single PUF

We propose Mixture-of-PUF-Experts layer (MoPE) on the basic MoE structure [20], as briefly outlined in Section II-D, for modelling PUFs. The structure of MoPE is depicted in Figure 4. The model accepts a challenge as input and produces the predicted response as output. Challenges are processed by an input layer connected to three experts. These experts are tailored to handle the distinct features of CRPs. Each expert comprises of two hidden layers, each with 32 neurons, called 'PUF Expert', which are designed for PUF modelling tasks. The first layer is directly connected to the input layer, while the second links to the gate function. The gate function

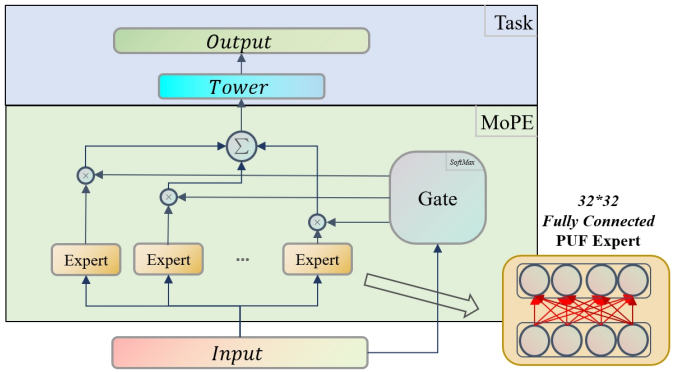


Fig. 4. Generic framework for modelling a single PUF.

assigns weights to the experts, amalgamates their outputs, and channels them to the tower. Initially, we convert the challenge bits $C^n = \{c_1, c_2, \ldots, c_n\}$ (where n represents the PUF stages) into the feature vector $X^n = \{x_1, x_2, \ldots, x_n\}$, aligning with the structure of delay-based PUF:

$$x_i = \prod_{j=i}^{n} c_j. \tag{4}$$

This transformation aids the model in perceiving the decision boundary as a hyperplane. The response r serves as the label and is adjusted to the range $[0, 1]$, if not already within it, to align with the activation function. Post feature engineering, the input layer is structured to accommodate these features. In the MoPE layer, we establish E experts, with the count being adaptable based on the number of tasks. The expert structure remains consistent across all PUF types, as two hidden layers equipped with non-linear activation functions are believed to model any function, given sufficient parameters. The $e-$th expert, denoted as $f_e(\cdot)$, is designed to extract specific insights or features from the input. Each expert delivers their unique interpretation of the input: $h_e(X) = f_e(X^n)$.

To harness the expertise of various experts without overburdening the model with excessive parameters, we introduce the gate function $g(x)$. This function evaluates the features and determines the weight. We employ the $softmax(\cdot)$ activation function post the $N \times E$ kernel W_{ge} to distribute weights among experts and ensure that the model prioritises the most apt one. Consequently, weights are computed as: $g(X) = softmax(W_{NE}(X))$. The weight assigned to the e-th expert is represented as $g^e(X)$, ensuring that $\sum_{e=1}^{E} g^e(X) = 1$. Here, we use multiple (5 for single-task, adaptive for multiple-tasks) same experts to ensure the success of modelling. However in most cases, the task will not use all the experts.

Now, to accelerate the training process, we propose a method called 'Sparse Softmax', as shown in **Algorithm 1**. Sparse Softmax function automatically sets the weights below a certain threshold τ to zeros, which has two benefits. First, the model pays more attention to the more suitable experts and helps accelerate the training. Second, the model can be flexible regarding scale size such that even overdoses of experts will not slow the training down too much, since the backward propagation will cost much more resources and training time than forward propagation, and zero weights need no calculations. The Sparse Softmax function is crucial for the generic framework, since it prevents the overfitting problem for PUFs with simple structures and help the convergence speed when modelling complex PUFs. Subsequently, the MoPE layer's output is derived by amalgamating the outputs of the experts: $mope(X) = \sum_{e=1}^{E} g^e(X) h_e(X)$. We then establish the tower layer, $T(\cdot)$, tasked with processing the composite information supplied by the experts. This layer then connects to the output layer, which employs the $sigmoid(\cdot)$ activation function to restrict the prediction output to the range $0, 1$. Our rationale for selecting a dual-layer hidden structure aligns with the perspective of Wisiol et al. in [13]. Viewing neural networks as a potent instrument for PUF modelling, we are confident that, given ample parameters and layers, PUFs can be effectively modelled, barring optimization constraints. Fewer layers expedite model convergence. Additionally, the MoPE structure's inherent flexibility allows the gate function to integrate multiple experts, facilitating network scalability to accommodate the diverse complexities inherent in PUFs.

Algorithm 1 Sparse Softmax Activation

Require: Input vector $\mathbf{W}_{\mathbf{NUM_{Experts}}}$, threshold $\tau = 0.0001$
Ensure: Sparse softmax vector $\widehat{\mathbf{W}}_{\mathbf{NUM_{Experts}}}$

```
procedure SparseSoftmax(W, τ)
    sum ← 0
    for j = 1 to NUM_Experts do
        sum ← sum + e^{W_j}
    end for
    for i = 1 to NUM_Experts do
        W_i ← e^{W_i} / sum
        if W_i < τ then
            W_i ← 0
        end if
    end for
    return Ŵ
end procedure
```

B. Generic Framework for Multiple PUFs

This section extends the framework to a multi-task learning model to enable the attack over multiple unique PUF instances. As discussed in Section II-A, the additive delay function represents the inner interaction of PUFs. We can find that they share similar mathematical formulations for the same category of PUF. Taking XOR-Arbiter-PUF as an example, for a k-XOR APUF, the responses can be represented as $\mathcal{P}(c) = \text{sgn}(\prod_{l=1}^{k} (\langle W_l, x \rangle + b_l))$, signifying that various modelling tasks exhibit commonalities as they employ an identical mathematical representation with varying parameter (delay) values. Here b_l is the bias value for the l-th PUF. Thus, a multi-tasks framework can help improve the modelling performance. The structure of the generic framework for multiple PUFs is shown in Figure 5.

The number of PUF experts can be customized according to the potential types and numbers of PUFs. We add three more PUF experts per extra task. The idea is that different

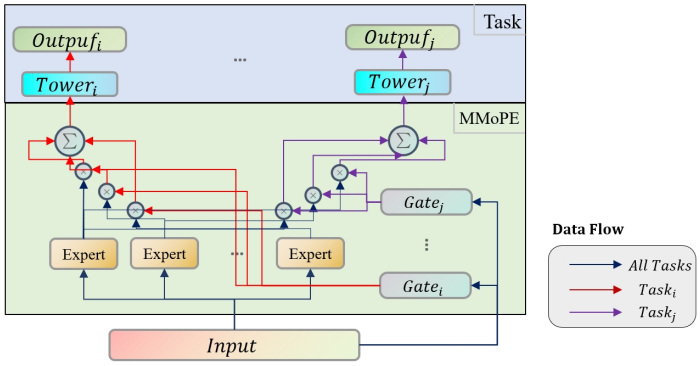


Fig. 5. Generic framework for modelling multiple PUFs.

structures or numbers of neurons in hidden layers are suitable for different types of PUFs. For some simple PUFs, if the model is too complex, it will be hard for the model to converge without a large amount of training data. This can also be concluded from the comparison results shown by Wisiol et al. in [13]. In this case, the gate function helps choose the suitable expert from the MoPE layer to alleviate optimization problems caused by too large a model. In Section III-A, we have already shown how to build the single-task model based on MoPE. For multiple tasks, we assign one gate function, tower function, and output layer to each task. In Figure 5, we give an example of modelling several PUFs simultaneously. In this figure, the red arrows represent dataflow for $Task_i$ and the purple ones for $Task_j$. The dark blue arrows are the common dataflow for both tasks. The feature engineering and input layer remain the same as described in Section III-A. The PUF expert structure is also the same, but the number of PUF experts may differ according to the number of PUFs.

There are N_{PUFs} gate functions, tower functions, and output layers for N_{PUFs} PUF models, so-called as N_{PUFs} tasks. For the t-th task, the weighed output of MoPE layer is:

$$moe^t(X) = \sum_{i=1}^{K} g_t^i(X) h_i(X). \tag{5}$$

Then, the specific tower $T_t(\cdot)$ for task t will deal with the extracted features provided by the experts and forward the results to the output layer to give the prediction.

C. Modelling Setup

We implement our method and replicate other methods using Python 3.8 and TensorFlow 2.4 [25] back-end executed on a Windows laptop with 48 GB of main memory, 5GHz i9-12900H Intel(R) Core processor and NVIDIA GeForce RTX 3070 Ti Laptop GPU. The proposed method should to be generic for any composition of Arbiter-based PUFs; thus, we set up one model and experimented with this model with the same settings. In Section III, we introduced the structure of the generic framework; we show the hyperparameters used in the experiments in Table I. We use $Relu$ as the activation function for all the hidden layers, $Softmax$ for the gate function and $Sigmoid$ for the tower function. For all kinds of PUFs, the

TABLE I
HYPERPARAMETER VALUE USED

Hyper Parameters	Values
Kernel Initializer	Glurot uniform
Optimizer	Adam[26]
Hid. Lay. activ.	Relu
Learning rate	Adaptive
Loss function	BCE

MoPE has four experts, each with two fully connected hidden layers with 32 neurons. We adjust the batch size according to the scale of the training dataset. With the number of CRPs denoted as N_{crp}, we set $batch\_size = min\{N_{crp}, 20000\}$. From the experiments, we find that dynamic adjustment can help the network to converge faster. All the codes, data and implementation details are presented in our anonymous GitHub repository[2].

D. Data Preparation

We conduct experiments on three unique datasets: one simulated dataset built by an additive-delay model, one collected from PUF designs synthesised in hardware, provided by Mursi et al. [11], and one biased dataset collected from a non-layout-optimized designed by us. Each dataset simply consists of CRPs. In our simulated dataset, the challenges are generated using the *PyPuf* [27] python library's random generator function as a set of binary strings of length n corresponding to the PUF challenge length. Responses are generated by applying each challenge to each tested PUF instance, each of which is generated with a random seed (for each unique PUF). The seed is randomly chosen and guaranteed to differ for every simulated data generated in this article. For more details, please refer to the code[2].

IV. GENERIC FRAMEWORK FOR MODELLING SINGLE PUF

In this section, we first show how to use the generic framework for any kind of delay-based PUF without knowing any extra information other than CRPs **and answer RQ 1 and RQ 2**. Then, we present the results and analysis based on the experiments. For our initial comparison, we present the experiment settings and results for the proposed generic framework of a single PUF.

A. Modelling Accuracy Results on Simulated Dataset

As shown in Table II, we present the performance of our method on modelling XOR PUF and compare them with other state-of-the-art machine learning models. Overall, our method successfully performs attacks on different structures of Arbiter-based PUFs without the need to change settings. With the benefit of a fixed structure, rather than a structure that expands as the PUF structure becomes more complex,

[2]**The code and datasets used in this paper are provided in full for the use of the research community at: https://github.com/AnonymousAppdx/Generic-Framework-for-Modelling-PUFs**

the training time does not increase exponentially with the complexity of PUFs. From the accuracy results, we can find that our method does not require more training CRPs to counteract the negative effects brought by the fixed network structure. During the experiments, the strategy for every method is to begin the modelling at a reasonable amount of CRPs; then, if the attack succeeds with accuracy beyond 90%, we attempt more random PUFs. If an attempt fails, we increase the amount of CRP training data; otherwise, if the method models all the PUFs, we decrease the amount of training data and continue the test. A detailed algorithm for finding the minimal amount of training data is shown in Algorithm 2.

The inputs are the attack model $\mathcal{A}$, challenge database $DTBS_C$, PUFs database $DTBS_{PUF}$, an initial value of minimal amount of CRPs m, and the epoch rounds n_{epoch}. The algorithm outputs the minimal amount of CRPs n_{mini}, the average accuracy Acc_{avg}, and average time T_{avg}. The whole algorithm is a **while** loop; it only breaks if the model $\mathcal{A}$ achieves accuracy beyond 90% for all the PUFs in $DTBS_{PUF}$ on n_{epoch} different challenges randomly selected from $DTBS_C$. After the two **for** loops, the average accuracy and time consumed is calculated, and the final amount of CRPs is recorded. Then, the **while** loop breaks. When a model, e.g., MoPE, is evaluated using **Algorithm** 2 against PUF PUF and outputs Acc_{avg}, T_{avg} and n_{mini}, we can say with at least n_{mini} CRPs, model $\mathcal{A}$ can break any PUF of this type in any situation. In other words, PUF is totally broken when n_{mini} CRPs are released.

The idea is to find the point where each attack succeeds enough to be *practically useful*, as described in Section II-G, against even fairly large PUFs, can achieve this consistently, and uses as few CRPs as possible within those constraints. In order to make a fair evaluation, we apply the strategy on all the methods presented in this section. While 90% is the point at which we consider a model practically useful, that is not to say higher accuracy would not be better. If improvement beyond 90% is possible without too much cost, this is still useful. For this reason, we employed a patience strategy, stopping after any 10 sequential epochs with minimal change in accuracy. The final results do not represent the true upper limit of these models; given sufficient time and resources, some could achieve up to 99%. However, improvements in the precision of even a few points above a certain threshold - typically 90-95% - require an expenditure that is several times greater than that needed to reach the threshold in the first place. The strategy we employed captures the level of performance that can be achieved before hitting this slowdown. From the adversary's perspective, pushing past this point is rarely worthwhile as they already have a successful, if sub-optimal, attack.

1) XOR Arbiter PUF: We mainly refer to two state-of-the-art neural network models [10], [11] and one logistic regression model [4] to compare our results. These models offer detailed guidance for implementing the codes that can be easily replicated. Additionally, their claimed accuracy is close to our observations. To make the comparisons fair and reasonable, we evaluate the performance of compared models by reproducing them using the same dataset and hardware resource. Besides our strategy for finding a stable amount of training data, we tried to find the best hyperparameters for compared schemes that were not disclosed, e.g., the training batch size.

Algorithm 2 Security Evaluation on Minimal Amount of CRPs

Require: model $\mathcal{A}$, challenge database $DTBS_C$, PUFs database $DTBS_{PUF}$, minimal amount m, n_{epoch}
Ensure: n_{mini}, Acc_{avg}, T_{avg}

procedure SECURITY EVALUATION($\mathcal{A}$,$DTBS_C$,$DTBS_{PUF}$,m,n_{epoch})
  while true **do**
    $Acc_{avg}, T_{avg} \leftarrow 0$
    for puf in $DTBS_{PUF}$ **do**
      for $i \leftarrow 1$ **to** n_{epoch} **do**
        $Cs \leftarrow$ select$(DTBS_C, m)$
        $Rs \leftarrow puf$.eval(Cs)
        $acc \leftarrow \mathcal{A}$.train_and_evaluate$(Cs, Rs, n_{epoch})$
        if $acc < 90\%$ **then**
          $m \uparrow\uparrow$
          break
        end if
        Acc_{avg} += acc
        T_{avg} += Δt
      end for
      if $acc < 90\%$ **then**
        $m \uparrow\uparrow$
      end if
    end for
    if $acc < 90\%$ **then**
      $m \uparrow\uparrow$
      continue
    end if
    $Acc_{avg} = \frac{Acc_{avg}}{NUM_{puf} \times n_{epoch}}$
    $T_{avg} = \frac{T_{avg}}{NUM_{puf} \times n_{epoch}}$
    $n_{mini} \leftarrow m$
  end while
end procedure

For the method presented by Rührmair et al. [4], we use code provided by Wisiol et al. in the PyPuf Python library [27]. This attack builds the network strictly according to the mathematical models. For small stages of XOR-APUFs, it outperforms all other methods in terms of training time (lower is better) and accuracy. As stated in [28] **Theorem 1**, the logistic regression method is the most powerful attack among classical machine learning attacks; however, as the number of stages increases, LR consumes the most training data and time. *Intuitively, when the number of XORs is more than 5, our proposed method has more advantages.*

For the attacks presented by Asseri et al. in [10] and Mursi et al. in [11], we implement their methods using the same setting claimed in their paper, including the kernel initializer, optimizer, learning rate, activation functions, and loss function. However, we optimize the batch size and adapt it according to the scale of training data. During the difficult reproducing work of different schemes, we find that the success of an attack

TABLE II
MODELLING RESULTS OF FOR SINGLE XOR PUF ON SIMULATED CRPS

Method	k	crp	time	acc	Memory
Rührmair et al[4]		8k	<20sec	>97.0%	1.78 GiB
Aseeri et al[10]	2	8k	<20sec	97.0%	1.79 GiB
Mursi et al[11]		8k	<1min	>98.0%	1.86 GiB
Proposed Scheme		8k	<20sec	>94.0%	2.06 GiB
Rührmair et al[4]		20k	<20sec	>99.0%	1.84 GiB
Aseeri et al[10]	3	24k	<20sec	>97.0%	1.81 GiB
Mursi et al[11]		24k	<1min	>98.0%	1.88 GiB
Proposed Scheme		24k	<1min	>95%	2.15 GiB
Rührmair et al[4]		30k	<20sec	>99.0%	1.91 GiB
Aseeri et al[10]	4	100k	<1min	>98.0%	1.91 GiB
Mursi et al[11]		120k	<1min	>98.0%	1.97 GiB
Proposed Scheme		80k	<1min	>97.0%	2.26 GiB
Rührmair et al[4]		260k	<20sec	>99.0%	1.88 GiB
Aseeri et al[10]	5	400k	<2min	>95.0%	2.32 GiB
Mursi et al[11]		240k	<1min	>98.0%	2.29 GiB
Proposed Scheme		240k	<1min	>97.0%	2.24 GiB
Rührmair et al[4]		3M	<1min	>99.0%	2.34 GiB
Aseeri et al[10]	6	1.6M	<2min	>99.0%	4.29 GiB
Mursi et al[11]		1.6M	<2min	>99.0%	3.04 GiB
Proposed Scheme		800k	<2min	>96.0%	3.31 GiB
Rührmair et al[4]		20M	<1hr	>98%	4.61 GiB
Aseeri et al[10]	7	5M	<20min	>97%	8.05 GiB
Mursi et al[11]		4M	<20min	>98.0%	8.07 GiB
Proposed Scheme		2.4M	<20min	>98.0%	5.08 GiB

highly relies on many factors: the dataset, the initial state of the model, the structure of the network, and even the batch size. Many ML-MA works indicate optimal accuracy and present their best accuracy on specific PUFs with few training CRPs. When we try to apply the method to a different dataset, it fails or can not achieve a reliable success rate. In some cases, different initialisations of the kernels succeed on some datasets and fail on others. The modelling process can be susceptible to differing hyperparameters. This kind of unreliable attack is unacceptable for realistic adversaries. Besides, the methods of Asseri et al. [10], and Mursi et al. [11] designed different structures for different PUFs. In most cases, they cannot learn the PUFs of different types of PUFs, which means they need to know the type information. We discerned their acute sensitivity to the PUF's structure through our attack implementations on these schemes. This underscores the indispensability of type/structure information for mounting successful attacks. For instance, in [11], the neural network devised for attacking a k-XOR Arbiter PUF comprises of three fully connected hidden layers sized $\{2^{k-1}, 2^k, 2^k\}$. This implies that for a 5-XOR APUF and 6-XOR APUF, the hidden layer structures should be $\{2^4, 2^5, 2^4\}$ and $\{2^5, 2^6, 2^5\}$, respectively. We validated their efficacy on the stated CRP quantities, 200k and 200M. Yet, when we experimented with the $\{2^4, 2^5, 2^4\}$-structure neural network for the 6-XOR APUF and the $\{2^5, 2^6, 2^5\}$-structure for the 5-XOR PUF, both attempts were unsuccessful. These outcomes underscore that non-generic models are meticulously crafted; simplistic models falter with complex PUFs, and conversely, intricate models struggle with simpler PUFs. Potential reasons could range from optimization challenges to under-fitting in the former scenario and over-fitting in the latter.

Next, we evaluate the generic capability of different models as shown in Table III, from where we can find that our proposed scheme can achieve good accuracy in modelling different PUFs. Table III shows the performance of different models, including MoPE, Mursi et al. [11], Aseeri et al. [10], Rührmair et al. [4] and Mishara et al. [14], on XOR-PUFs ranging from 2 to 7 XORs. The sub-columns of each method represent the model designed for the specific type of PUF. For example, the cell positioned at {3-XOR, Mursi [11], 2} indicates that the accuracy of modelling 3-XOR-PUF using 2-XOR-PUF-Model is 98%. The table shows that our proposed model can perform the modelling attack across any XOR-PUF with accuracy ranging from $94 - 98\%$. On the other hand, this kind of cross-architectural modelling capability has not been considered in all other methods. Consequently, this leads to a pertinent question: How can we compare the security levels of two distinct PUFs? Furthermore, how should we select the model and conduct the evaluations? For instance, if we consider the 7-XOR-PUF model proposed by Aseeri et al. and evaluate the 6-XOR-PUF and 5-XOR-PUF, we get the accuracy of 98% and 50%, which denotes that the security-level of 5-XOR-PUF is stronger than 6-XOR-PUF, which is not true. In a nutshell, if we consider evaluating the security performance of a structure-unknown PUF or comparing distinct PUFs, our proposed model has a significant advantage over all other methods.

2) XOR Feed-Forward Arbiter PUF and Interpose PUF: In this section, we demonstrate the versatility of our model in adapting to a range of modelling tasks. Specifically, we apply our generic model to various additional delay-based PUFs, encompassing even those PUFs previously resistant to successful attacks. Avvaru et al. introduced the homogeneous and heterogeneous Feed-Forward XOR PUFs in [29]. Subsequent to their work, a multitude of machine learning models were proposed to target FF-APUFs [13]. A large portion of these models capitalize on the inconsistent reliability of PUF designs, focusing particularly on homogeneous XOR FF PUFs with uniform loop positions. In contrast, heterogeneous XOR-FF-APUFs are largely considered resilient against modelling attacks. As evidenced in Figure IV, we successfully modelled $2-$loop FF PUFs with 2 XOR stages and $1-$loop FF PUFs with 3 XOR stages, achieving accuracy exceeding 95% and 98%, respectively. The Interpose PUF was introduced by Nguyen et al. in [28]. Although it was later targeted using the 'divide-and-conquer' technique [30], our results, as depicted in Figure IV, confirm that our proposed model can adeptly launch successful attacks on various configurations of the Interpose PUF without necessitating any structural alterations to the model itself.

B. Modelling Accuracy Results on Silicon CRPs

To avoid faulty evaluations caused by wrongly generated simulated data, we validate our method on both unbiased real-world data provided by Mursi et al. [11] and biased implementations. We include the biased dataset primarily to demonstrate the profound impact even minimal (and potentially otherwise accepted) amounts of bias can have in providing knowledge to

TABLE III
COMPARISON OF GENERIC MODELLING CAPABILITY

Method	MoPE	Mursi[11]						Aseeri[10]						Rührmair[4]						Mishra[14]				
PUFs/Model	Ours	2	3	4	5	6	7	2	3	4	5	6	7	2	3	4	5	6	7	1	2	3	4	5
2-XOR	94	98	98	98	98	95	95	98	97	96	93	92	93	99	97	99	59	50	52	75	-	-	-	-
3-XOR	95	98	98	98	98	96	95	98	97	97	95	92	87	66	99	99	99	50	54	-	78	-	-	-
4-XOR	97	98	98	98	98	98	51	50	50	98	50	56	55	60	63	99	50	99	52	-	-	77	-	-
5-XOR	97	98	98	98	98	99	50	50	50	50	95	50	50	55	58	52	99	50	50	-	-	-	80	-
6-XOR	96	50	50	50	50	99	50	50	50	50	50	99	98	50	54	59	67	99	50	-	-	-	-	79
7-XOR	98	50	50	50	50	50	98	50	50	50	50	50	97	50	50	50	50	50	99	-	-	-	-	-

TABLE IV
MODELLING RESULTS OF FOR FEED-FORWARD PUFS AND INTERPOSE PUFS.

Type	k	Loops	crp	time	acc
Homogeneous FF-PUF	1	1	20k	<2min	>94%
		2	120k	<2min	>97%
		3	250k	<2min	>98%
		4	500k	<2min	>98%
		5	1M	<10min	>94%
	2	1	90k	<2min	>93%
		2	200k	<10min	>97%
		3	400k	<10min	>93%
		4	800k	<20m	>85%
		5	1.6M	<1hr	>90%
	3	1	120k	<2min	>90%
		2	400k	<10min	>93%
Heterogeneous FF-PUF	2	1	160k	<2min	>98%
	3	1	640k	<10min	>98%
	2	2	400k	<2min	>95%
Interpose PUF	**Upper chains**	**Lower chains**	**crp**	**time**	**acc**
	1	5	480k	<2min	>98%
	3	3	320k	<3min	>98%
	4	4	800k	<1hr	>96%
	5	5	2M	<1hr	>96%

TABLE V
MODELLING RESULTS OF FOR SINGLE XOR PUF ON SILICON CRPS, FROM [11] AND OUR NON-OPTIMIZED IMPLEMENTATIONS.

	Mursi et al. [11]			**Our data**		
k	**crp**	**time**	**acc**	**crp**	**time**	**acc**
4	40k	<20sec	92.56%	40k	<20sec	92.90%
5	160k	<20sec	95.21%	120k	<1min	95.36%
6	560K	<1min	94.56%	120k	<1min	95.36%
7	1.6M	<1min	93.38%	600k	<10min	95.81%

the adversary's MLA. We randomly select the demand amount of CRPs from the silicon data and apply our framework without changing any model settings, which is needed for all other models. The results are shown in Table V. We can find that the accuracy does not decrease at all for any stage of XOR-APUFs when we use the same amount of CRPs.

Then, we implement 64-stage Arbiter PUFs on a Zynq-7000 FPGA using the Vivado design suite. Verilog hardware description language is used to build the PUF design. No placement design is applied to these PUFs. The hardware layout is shown in Figure 6. We applied a DRAM controller to transfer challenges and obtain responses between the programmable logic (PL) and processing subsystem (PS) sides. For each single Arbiter PUF, we evaluate its performance.

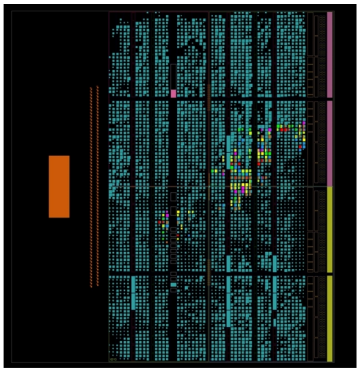
TABLE VI
MODELLING RESULTS OF FOR MULTIPLE (**TWO**) XOR PUFS ON SIMULATED CRPS.

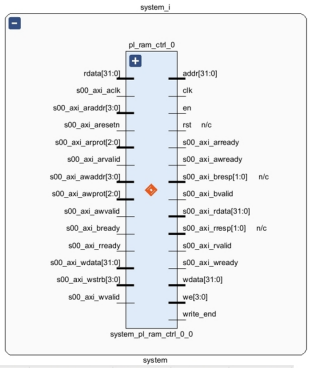
Method	k	crp	time	acc
Aseeri et al[10]	2	10k	<20sec	71.5%
Mursi et al[11]		10k	<20sec	67.34%
Proposed Scheme		8k	<20sec	93.50%
Aseeri et al[10]	3	30k	<20sec	72%
Mursi et al[11]		30k	<20sec	74.39%
Proposed Scheme		24k	<20sec	93.02%
Aseeri et al[10]	4	100k	<20sec	73.72%
Mursi et al[11]		100k	<20sec	74.60%
Proposed Scheme		80k	<20sec	93.99%
Aseeri et al[10]	5	400k	<20sec	50%
Mursi et al[11]		400k	<20sec	74.89%
Proposed Scheme		240k	<20sec	97.88%
Aseeri et al[10]	6	2M	<1min	74.08%
Mursi et al[11]		2M	<1min	74.08%
Proposed Scheme		800k	<20sec	95.04%
Aseeri et al[10]	7	5M	<1hr	74.20%
Mursi et al[11]		5M	<20min	50%
Proposed Scheme		2.4M	<20min	98.04%

The average bias is around 55%, which makes the PUF more vulnerable to ML-MA. As shown in Table V, much fewer CRPs are needed to perform successful attacks compared to unbiased implementations.

C. Resource Consumption and Efficiency Analysis

In many modelling attack works targeting PUFs [13], [16], [11], [10], the structure of the model varies for different PUFs. As our framework is designed for a generic purpose, it is not suitable to compare the scale of models directly for consumption evaluations. However, we list the time cost and memory usage in Table II. The time cost of our proposed scheme is not much different from other compared methods and outperforms Rührmair et al [4] and Aseeri et al. [10] when attacking 7-XOR APUF. We used more memory in our scheme for attacking, which is attributed to the number of neurons and layers that are required in our model. For 7-XOR APUF, 5.08 GiB memory is needed, and methods in [4], [10], [11] need 4.61, 8.05, 8.07 GiB. We clarify here that our memory usage might be different than that claimed by the authors [4], [10], [11]; in addition to the hardware differences, we used larger batch sizes to speed up the training. For our model, the memory usage between different tasks mainly depends on the size of the dataset, which determines the batch size. Overall, we fixed the structure for all types of PUFs and the most





Name	Slice LUTs (17600)	Slice Registers (35200)	F7 Muxes (8800)	F8 Muxes (4400)	Block RAM Tile (60)
system_wrapper	9832	11170	949	469	2
system_i (system)	9832	11170	949	469	2
axi_bram_ctrl_0 (system_axi_bram_ctrl_0_0)	155	166	0	0	0
axi_gpio_0 (system_axi_gpio_0_0)	197	449	0	0	0
axi_intc_0 (system_axi_intc_0_0)	69	64	0	0	0
axi_smc (system_axi_smc_0)	5710	5655	4	0	0
blk_mem_gen_0 (system_blk_mem_gen_0_0)	0	0	0	0	2
pl_ram_ctrl_0 (system_pl_ram_ctrl_0_0)	3663	4811	945	469	0
processing_system7_0 (system_processing_system7_0_0)	24	0	0	0	0
rst_ps7_0_50M (system_rst_ps7_0_50M_0)	14	25	0	0	0

Fig. 6. Hardware layout, schematic and overhead for 7 PUFs on XC7Z010 FPGA Board. No optimization design for the layout was performed.

complex PUF determines the hardware requirement for our model.

V. Generic Framework of Multiple PUFs

In Section IV, we have shown the generality of the proposed model that does not need to design specific structures for different types of PUFs, and no information apart from CRPs is needed. Next we evaluate another generic property of our proposed model on multiple tasks. To the best of our knowledge, multi-task learning for multiple PUFs has not been studied before. In this section, we first show how multi-task learning challenges the existing modelling methods [4], [10], [11] and compare them with ours. Then, we present our modelling accuracy results on different combinations of various types of PUFs.

A. Setup

In Section III, we showed the structure of a generic framework for multi-task learning and the strategy of choosing experts. In the experiments, we first set up several randomly chosen PUFs, then generated the challenges and used them to query all the PUFs and collect the responses to store in a list. We feed the challenges and the response list to the model. The model does not know from which PUF the data is from but creates one gate function, one tower layer, and one output layer per group of responses. The model is expected to predict all the responses to unseen challenges for all the PUF inputs.

B. Modelling Results Comparison and Summary

We employed extra output layers on contrasting schemes [10], [11] to enable their multi-task learning capability. After the modifications, their models look like the share-bottoms [18], which is the most common way of enabling multi-task learning. *For [4], we cannot add output layers for a mathematical-model-based structure since they used multiplying layers and directly output the result after the activation function.* Due to the new methodology proposed in [14], its publication shortly before the time of writing, and some issues with the released codebase (also reported in [31]), its replication was not feasible within the scope of this work. Instead, We have referenced the provided results in [14] for comparison purposes, though this is limited by what tests were performed in the work referenced. We add one extra output layer for [11], [10]. The results for modelling multiple XOR PUFs are shown in Table VI. The results show that applying multi-task learning features on a proven feasible model cannot balance between different tasks and will always fail one random task, described as the result of "Negative Transfer" when the two tasks have low similarity or the experts cannot understand the correlation. In this case, multi-task learning does not work and even the performance for training the single task is degraded by the other conflicting task. However, as shown in Table VI, our proposed model can deal with different tasks with good performance compared to the same single task. For two 7-XOR APUFs, we need the same amount of CRPs for both and achieve an average accuracy of 98.04% in 20 minutes. On average, we can save half of the training time, which will be efficient for an adversary modelling multiple PUFs. We also list more combinations of different numbers of various PUFs in Table VII to show the flexibility of the proposed model. Specifically, we are testing the modelling ability for multiple PUFs *where the PUFs are not all of the same type* e.g., some devices being attacked have FF-APUFs and others have Interpose PUFs, devices use XOR-APUFs of varying size, etc. We test several such combinations to demonstrate the consistency of the method but not every possible permutation to keep this experiment's complexity manageable. In particular, in comb. I, we show the results of modelling four XOR APUFs with the same stages and different stages. In comb. II, we show the results of modelling four homogeneous XOR FF-APUFs with different types. In comb. III, we show the results of modelling four heterogeneous XOR FF-APUFs. In comb. IV, we show the results for different combinations of XOR APUFs, (two kinds) XOR FF-APUFs, and Interpose PUFs. Compared to the results in Table II, 3% of accuracy is traded off on average, and the maximum is around 8%. This shows that even for systems containing mixtures of completely different PUF designs, our approach consistently achieves viable attack capability (at least, for cases where all are delay-based PUFs).

VI. Conclusion

In this study, we proposed a generic framework for modelling different delay-based PUFs. In this regard, we introduced a new notion called the Mixture-of-PUF-Experts layer that enables attacks with minimal knowledge using the gate function and experts of PUFs. A realistic threat model has been considered where the Minimum Viable Adversary can only sniff the data in the network without knowing any other

TABLE VII
RESULTS OF ATTACKING DIFFERENT COMBINATIONS OF DIFFERENT PUFS WITH OUR PROPOSED SCHEME

Comb. No.	Combinations	Numbers of PUFs	Performance	
			Average Accuracy	Running time
I	4 × 3-**XOR APUF**†	4	>93%	<2min
	4 × 4-**XOR APUF**†	4	>93%	<10min
	5 × 3-**XOR APUF**†	4	>94%	<20min
	2 × 3-**XOR APUF**†, 2 × 4-**XOR APUF**†,1 × 5-**XOR APUF**†	5	>92%	<10min
II	4 × (2-1)-**Homo. XOR FF-APUF**††	4	>95%	<20min
	4 × (1-3)-**Homo. XOR FF-APUF**††	4	>92%	<30min
III	4× (3-1)- **Hete. XOR FF-APUF**††	4	>92%	<30min
	4× (2-2)- **Hete. XOR FF-APUF**††	4	>95%	<20min
IV	4 × (1,5)-**Interpose PUF**‡	4	>96%	<30min
	2 × 3-**XOR APUF**†, 2 × 4-**XOR APUF**†,1 × (1,5)-**Interpose PUF**‡	5	>96%	<30min
	1 × (2-1)-**Hete. XOR FF-APUF**,1 × (2-2)-**Hete. XOR FF-APUF**,1 × (3-1)- **Hete. XOR FF-APUF**††,1 × (1,5)-**Interpose PUF**‡	4	>90%	<30min
	1 × (1,5)-**Interpose PUF**‡,1 × (2,2)-**Interpose PUF**‡,1 × (3,3)-**Interpose PUF**‡	3	>90%	<30min

† The parameter indicates the number of XOR stages;
†† The two parameters indicate the number of XOR stages and loops;
‡ The parameter indicates the number of parallel arbiter chains in up and lower layers.
* In the table, the amount of training data for each PUF is the same as the corresponding single task.

information about the communication objects. We showed successful attack results on XOR-APUFs, both homogeneous and heterogeneous XOR FF-PUFs, and Interpose PUFs, without changing any settings of the model which answers **RQ 1**. Besides, we have also proposed an extended version of MoPE i.e., Multi-Gate Mixture-of-PUF-Experts. It enables multi-task modelling on PUFs, which can capture the relationship between similar PUFs and accelerate the modelling process and answer **RQ 3**. We are the first to enable multiple-PUF attack capability of adversaries without incurring an unacceptable loss of accuracy. To facilitate a fair comparison with the latest advancements in PUF modelling, we have undertaken analogous experiments to those conducted in previous studies, such as [4], [11], [10]. Experiments on different datasets, including simulated, biased silicon and unbiased silicon data, were performed to validate our methods. We argue that our proposed models successfully solve **RQ 2** and will be helpful for the PUF community, especially when one comes up with a new PUF design and is willing to test whether their PUF is ML-MA secure without disclosing details of the PUF. In this research, we mainly considered the delay-based PUF(s); however, in the future, we would like to consider other categories of PUFs.

ACKNOWLEDGMENT

This research/project is supported by the National Research Foundation, Singapore and Infocomm Media Development Authority under its Future Communications Research and Development Programme under project FCP-NUS-TG-2022-001. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of the National Research Foundation, Singapore and Infocomm Media Development Authority. The work of Prosanta Gope was supported by The Royal Society Research Grant under grant RGS\R1\221183. The work of Jack Miskelly is supported by the Innovate UK SCHEME Project (IUK10065634).

REFERENCES

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

[2] A. Tria and H. Choukri, *Invasive Attacks*. Boston, MA: Springer US, 2011, pp. 623–629. [Online]. Available: https://doi.org/10.1007/978-1-4419-5906-5_511

[3] J. Shi, Y. Lu, and J. Zhang, "Approximation attacks on strong pufs," *IEEE transactions on computer-aided design of integrated circuits and systems*, vol. 39, no. 10, pp. 2138–2151, 2019.

[4] U. Rührmair, F. Sehnke, J. Sölter, G. Dror, S. Devadas, and J. Schmidhuber, "Modeling attacks on physical unclonable functions," in *Proceedings of the 17th ACM conference on Computer and communications security*, 2010, Conference Proceedings, pp. 237–249.

[5] P. Santikellur, A. Bhattacharyay, and R. S. Chakraborty, "Deep learning based model building attacks on arbiter puf compositions," *IACR Cryptol. ePrint Arch.*, vol. 2019, p. 566, 2019. [Online]. Available: https://eprint.iacr.org/2019/566.pdf

[6] J.-L. Lanet and C. Toma, *Innovative Security Solutions for Information Technology and Communications.* Springer, Cham, 2018.

[7] U. Rührmair, J. Sölter, F. Sehnke, X. Xu, A. Mahmoud, V. Stoyanova, G. Dror, J. Schmidhuber, W. Burleson, and S. Devadas, "Puf modeling attacks on simulated and silicon data," *IEEE transactions on information forensics and security*, vol. 8, no. 11, pp. 1876–1891, 2013.

[8] G. T. Becker, "The gap between promise and reality: On the insecurity of xor arbiter pufs," in *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer, 2015, Conference Proceedings, pp. 535–555.

[9] J. Shi, Y. Lu, and J. Zhang, "Approximation attacks on strong pufs," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 10, pp. 2138–2151, 2020.

[10] A. O. Aseeri, Y. Zhuang, and M. S. Alkatheiri, "A machine learning-based security vulnerability study on xor pufs for resource-constraint internet of things," in *2018 IEEE International Congress on Internet of Things (ICIOT)*. IEEE, 2018, pp. 49–56.

[11] 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," *Electronics*, vol. 9, no. 10, p. 1715, 2020.

[12] H. Fei, G. Prosanta, M. Owen, and S. Biplab, "Optimal machine-learning attacks on hybrid pufs," in *29th European Symposium on Research in Computer Security*, 2024, Conference Proceedings.

[13] N. Wisiol, B. Thapaliya, K. T. Mursi, J.-P. Seifert, and Y. Zhuang, "Neural network modeling attacks on arbiter-puf-based designs," *IEEE Transactions on Information Forensics and Security*, vol. 17, pp. 2719–2731, 2022.

[14] N. Mishra, K. Pratihar, S. Mandal, A. Chakraborty, U. Rührmair, and D. Mukhopadhyay, "Calypso: An enhanced search optimization based framework to model delay-based pufs," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2024, no. 1, pp. 501–526, 2024.

[15] W. Liu, Y. Zhang, Y. Tang, H. Wang, and Q. Wei, "Alsca: A framework for using auxiliary learning side-channel attacks to model pufs," *IEEE Transactions on Information Forensics and Security*, vol. 18, pp. 804–817, 2022.
[16] P. Santikellur, A. Bhattacharyay, and R. S. Chakraborty, "Deep learning based model building attacks on arbiter puf compositions," *Cryptology ePrint Archive*, 2019.
[17] 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)*, 2019, pp. 204–209.
[18] R. Caruana, "Multitask learning," *Machine learning*, vol. 28, pp. 41–75, 1997.
[19] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton, "Adaptive mixtures of local experts," *Neural computation*, vol. 3, no. 1, pp. 79–87, 1991.
[20] S. Wang, Y. Li, H. Li, T. Zhu, Z. Li, and W. Ou, "Multi-task learning with calibrated mixture of insightful experts," in *2022 IEEE 38th International Conference on Data Engineering (ICDE)*. IEEE, 2022, Conference Proceedings, pp. 3307–3319.
[21] A. D. Singh, L. Garcia, J. Noor, and M. Srivastava, "I always feel like somebody's sensing me! a framework to detect, identify, and localize clandestine wireless sensors," in *30th USENIX Security Symposium (USENIX Security 21)*, 2021, pp. 1829–1846.
[22] J. Zhang, C. Shen, Z. Guo, Q. Wu, and W. Chang, "CT-PUF: Configurable Tristate PUF Against Machine Learning Attacks for IoT Security," *IEEE Internet of Things Journal*, vol. 9, no. 16, pp. 14 452–14 462, 2022.
[23] C. Gu, C.-H. Chang, W. Liu, S. Yu, Y. Wang, and M. O'Neill, "A modeling attack resistant deception technique for securing lightweight-puf-based authentication," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 40, no. 6, pp. 1183–1196, 2020.
[24] P. Santikellur, S. R. Prakash, R. S. Chakraborty *et al.*, "A computationally efficient tensor regression network based modeling attack on xor apuf," in *2019 Asian Hardware Oriented Security and Trust Symposium (AsianHOST)*. IEEE, 2019, pp. 1–6.
[25] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin *et al.*, "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," *arXiv preprint arXiv:1603.04467*, 2016.
[26] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.
[27] N. Wisiol, C. Gräbnitz, C. Mühl, B. Zengin, T. Soroceanu, N. Pirnay, K. T. Mursi, and A. Baliuka, "pypuf: Cryptanalysis of physically unclonable functions," 2021.
[28] P. H. Nguyen, D. P. Sahoo, C. Jin, K. Mahmood, U. Rührmair, and M. van Dijk, "The interpose puf: Secure puf design against state-of-the-art machine learning attacks," *Cryptology ePrint Archive*, 2018.
[29] S. S. Avvaru, Z. Zeng, and K. K. Parhi, "Homogeneous and heterogeneous feed-forward xor physical unclonable functions," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 2485–2498, 2020.
[30] N. Wisiol, C. Mühl, N. Pirnay, P. H. Nguyen, M. Margraf, J.-P. Seifert, M. van Dijk, and U. Rührmair, "Splitting the interpose puf: A novel modeling attack strategy," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pp. 97–120, 2020.
[31] G. Li, "Could anyone reproduce the claimed result?" GitHub issue, Dec 2023, issue number: 1. [Online]. Available: https://github.com/SEAL-IIT-KGP/calypso/issues/1

Fei Hongming received the B.S. and M.S. degree from Northwestern Polytechnical University, Xi'an, China, in 2019 and 2022. He is currently pursuing a PhD degree in the Department of Electrical and Computer Engineering at the National University of Singapore. His research interests include. PUF-based security systems, PUF ageing, machine learning modelling attacks on PUFs, device fingerprinting, and secure protocol designs.

Prosanta Gope (Senior Member, IEEE) is currently working as an Associate Professor in the Department of Computer Science (Cyber Security) at the University of Sheffield, UK. Dr Gope served as a Research Fellow in the Department of Computer Science at the National University of Singapore (NUS). Primarily driven by tackling challenging real-world security problems, he has expertise in Lightweight Authentication, Authenticated Encryption, 5G and Next Generation Communication Security, Privacy-Preserving Machine Learning, Security in the Internet of Things, Smart-Grid Security, PUF-based security system and IoT Hardware. He has authored more than 100 peer-reviewed articles in several reputable international journals and conferences and has four filed patents. Several of his papers have been published in high-impact journals (such as IEEE TIFS, IEEE TDSC, IEEE TIE, IEEE TSG, and IEEE/ACM TON), and prominent security conferences (such as ACM CCS, IEEE Computer Security Foundations Symposium (CSF), Privacy Enhancing Technologies Symposium (PETS), ESORICS, Euro S&P, IEEE TrustCom, IEEE HoST, etc.) Dr Gope has served as a TPC member/Co-Chair in several reputable international conferences such as IEEE TrustCom, IEEE GLOBECOM(Security-Track), ARES, ESORICS, etc. He currently serves as an Associate Editor of the IEEE Transactions on Services Computing, IEEE Systems Journal, and the Journal of Information Security and Applications (Elsevier). His research has been funded by EPSRC, Innovate UK, and the Royal Society.

Biplab Sikdar (Senior Member, IEEE) received the B.Tech. degree in electronics and communication engineering from North Eastern Hill University, Shillong, India, in 1996, the M.Tech. degree in electrical engineering from the Indian Institute of Technology Kanpur, Kanpur, India, in 1998, and the Ph.D. degree in electrical engineering from the Rensselaer Polytechnic Institute, Troy, NY, USA, in 2001. He was a Faculty with the Rensselaer Polytechnic Institute, from 2001 to 2013, an Assistant Professor and an Associate Professor. He is currently a Professor and Head of Department of the Department of Electrical and Computer Engineering, National University of Singapore, Singapore. He also serves as the Director of the Cisco-NUS corporate Research Laboratory. His current research interests include wireless networks, and security for Internet of Things and cyber physical systems. He has served as an Associate Editor for the IEEE Transactions on Communications, IEEE Transactions on Mobile Computing, IEEE Internet of Things Journal and IEEE Open Journal of Vehicular Technology.