Collusion-resistant PUF-based Distributed Device Authentication Protocol for Internet of Things

Wassila Lalouani*, Mohamed Younis**, Mohammad Ebrahimabadi** and Naghmeh Karimi**
*Department of Computer and Information Science, Towson University, USA
**Department of Computer Science and Electrical Engineering, University of Maryland, Baltimore County, Baltimore, Maryland, USA,
Emails: wlalouani@towson.edu, {younis, ebrahimabadi, nkarimi}@umbc.edu

GLOBECOM 2022 - 2022 IEEE Global Communications Conference | 978-1-6654-3540-6/22/$31.00 ©2022 IEEE | DOI: 10.1109/GLOBECOM48099.2022.10001256

***Abstract*—The scale, unattended-operation and ad-hoc nature of an Internet-of-Things (IoT) make the network vulnerable to device impersonation, message replay, and Sybil attacks by either external actors or compromised nodes. This paper opts to tackle such vulnerability and presents a novel and effective solution for mutual authentication of IoT nodes. The proposed solution calls for embedding a Physically Unclonable Function (PUF) on each device, and employs a lightweight protocol for validating the identity of the individual devices based on querying the PUF. To authenticate a "prover" node, a verifier node will send a challenge bit-stream to the prover, where the latter provides the response of its PUF to such a challenge to be matched by what the verifier expects. To prevent the PUF of a prover from being modeled by an eavesdropper or a collusive set of compromised verifiers, the proposed protocol makes the response to a challenge dependent on the verifier. In addition, our protocol combines such an identity-based response generation with a simple Elliptic curve to thwart any attempts by a compromised verifier to reverse engineer the response generation process. The robustness of our PUF-based IoT Device Authentication (PIDA) protocol, is validated using data collected from an FPGA-based implementation.**

Keywords—IoT, Authentication, Physically Unclonable Function, Collusion resistance, Distributed security solution.

I. INTRODUCTION

The Internet of Things (IoT) is characterized by dynamic device membership, evolving network topology, and resource-constrained devices. These characteristics make decentralized management to be the preferred option since access to a central server cannot be ensured at all times. Moreover, the openness and pervasiveness of the IoT network raise major security concerns [1]. Particularly, device authentication is important given the major threat that infiltrating the IoT network would pose. In essence, impersonating a legitimate device can be the preferred means for an adversary to violate privacy and inject false data. Hence, a number of lightweight protocols are developed for establishing mutual trust among nodes [2]. However, many of these protocols are based on shared device secrets and do not withstand attacks that involve device hacking. A notable class among the published schemes rely on PUFs that are embedded in the IoT devices during manufacturing. PUF-based authentication protocols avoid storage of device secrets and instead generate them on-the-fly [3]. A PUF fundamentally realizes a one-way hash that is indexed by a set of bits, called challenge. The corresponding entry for a challenge is referred to as a response, and is not stored but rather is generated every time the challenge is applied to the PUF. Thus, PUFs are deemed tamper-proof primitives that enable resilience against device hacking.

To conduct mutual authentication of devices, existing PUF-based protocols rely on a central trusted server, which may not be accessible at all times. In fact, decentralized management is the preferred option in IoT. Therefore, distributed PUF-based device authentication methods are more appropriate. However, sharing challenge-response pairs (CRPs) among devices needs to be orchestrated such that the PUF cannot be modeled if multiple nodes collude. To elaborate, when a server is involved, each device D_p would provide a set of CRPs to be stored at the server. The latter uses these CRPs to authenticate D_p by sending one of the challenges and matches the response of D_p to what is stored at the server. Given that the server is trusted, the stored CRPs are not available to adversaries and cannot be used to develop a machine learning (ML) model that mimics the PUF [4]. On the other hand, when distributed authentication is to be pursued, a device D_p, referred to as a prover, has to share a set of CPRs with each communicating party, D_q, referred to as a verifier. Unlike the server, the verifier cannot be trusted and the CRPs of D_p could be leaked by D_q or used to model the PUF of D_p. The threat becomes even more serious if multiple verifiers collude, leading to the formation of an accurate model for D_p.

This paper fills the technical gap and presents PIDA, a novel distributed protocol for mutual authentication of IoT devices. Each device D_p will have an embedded PUF; yet the shared CPRs will be verifier-dependent, meaning that D_p will change the response of the same challenge based on the identity of D_q. Thus, even if multiple verifiers collude, they cannot model the PUF of D_p. Such verifier-dependent obfuscation of the PUF response also factors in a verifier-picked challenge bit-stream, and hence will thwart attempts to replay the prover's response to prior authentication requests. To generate the obfuscated response, PIDA employs a simple Elliptic-Curve Function (ECF), whose parameters are only known to the prover. The inputs to ECF are the binary responses of: (i) the PUF for the verifier-provided challenge, i.e., R_C=$PUF_p(C)$, and (ii) when using a verifier ID as a challenge, i.e., R_{ID}=$PUF_p(ID_q)$. We show that PIDA cannot be reversed engineered or modeled to predict the prover's response for unseen challenges. PIDA also mitigates the noise effect on the PUF output by storing the error-correction code (ECC) of the prover's binary response to a challenge at the verifier. The ECC will be provided by the verifier, along with the challenge, to help the prover tolerate noisy PUF outputs. Given the decorrelation between the PUF's binary output for the challenge and the actual obfuscated response used for authentication, sharing the ECC does not constitute much of information leakage. The validation results based on data collected from FPGA-based PUF implementation confirm the robustness of PIDA.

II. Related Work

Device authentication using asymmetric crypto-systems, imposes significant overhead and does not suit resource-constrained IoT devices [5]. Non-volatile memories such as EEPROM or battery-backed SRAM to store shared keys are not secure either [5]. Meanwhile, employing a trusted platform module increases the hardware complexity and is geared for software integrity rather than device authentication [6]. ML-based trust models are also pursued as a means to continually authenticate IoT nodes [7]; yet the associated computational overhead is high. Consequently, the use of PUF-based signatures has attracted attention in recent years [3]. However, most existing PUF-based authentication protocols rely on a centralized server [8], or are vulnerable to modeling attacks [4].

To counter modeling attacks, hardware-, encryption-, and protocol-based schemes have been pursued. In [9] the output is a function of the PUF response and the first and last challenge bits. A random number generator is used in [10] to shuffle the challenge and response bits. Yet, this approach requires synchronization unless the sequence of random numbers is predetermined. A circuit is added in [11] to shuffle the challenge bits in a manner that is dependent on the verifier ID. The shuffling process varies also based on the challenge bit patterns. Instead of shuffling, a secondary PUF is employed to obfuscate the challenge of the main strong PUF in [12]. To mislead the adversary, a fake PUF is deployed along with the original PUF [13]. However, the extra PUF and/or the obfuscation circuit impose significant overhead. Cryptographic hash functions are used in [14] to encrypt the PUF response. However, these schemes lose the PUF advantage by imposing significant overhead. Some protocol-level PUF modeling countermeasures pursue multifactor authentication. For example, the wireless channel characteristics are factored in [15]. However, channel noise variation could in fact hinder successful authentication. The approach of [16] uses password, PUF and biometrics, which impose high overhead; also, storing passwords defies the main advantage of PUFs.

Barbareschi et al. [17] use predefined chains of CRPs, where only the XOR values of the responses are sent; yet this scheme is vulnerable to impersonation attacks. In [18], multiple challenges are used where a function of the response is provided to the verifier. The goal is to counter eavesdropping threat; yet the approach is prone to collusion if applied in a distributed manner. Although T2T-MAP [19] is a server-based approach, the PUF responses are not stored at the server. In fact, the server can be an IoT node itself. T2T-MAP also avoids storing responses on the verifier; instead it creates entries that involve the PUFs of the two communication devices. Basically at enrollment, each device D_1 creates an authentication token that is unique for communication with another device D_2. However, the practicality of the approach is questionable given its inability to mitigate the PUF noise effect. Contrarily to most existing PUF-based schemes, PIDA enables collusion-resistant distributed authentication of IoT devices.

III. System Model and Approach Overview

A. System Model

The authentication process in IoT devices should be lightweight given the limited computation and communication resources, the scale of the network, and the dynamic connectivity where a device needs to be authenticated frequently. PIDA calls for embedding a PUF in each IoT device. A PUF is a hardware fingerprinting primitive that leverages random variations in the fabrication process of integrated circuits. A PUF simply maps an n-bit input stream, C, referred to as a challenge, to an output bit, referred to as a response. Since the process variations are random and independent of the devices, a PUF cannot be cloned and its challenge-to-response mapping constitutes a device signature. Fig. 1 shows the schematic diagram of the arbiter PUF, which is one of the prominent PUF types. Fundamentally the arbiter PUF relies on the variability of the delay experienced by an input signal until reaching the arbiter (can be realized as an SR-latch), where the propagation path is determined by the combination of challenge bits, i.e., c_0, c_1, …, c_{n-1}, that configure the multiplexers. To generate an m-bit response, either the PUF is queried by multiple challenges, e.g., arbitrary range of binary bit strings, or the PUF circuit is replicated m times. In this paper, we generally use R to reflect an m-bit response.

A PUF is an attractive choice for supporting authentication since the response of challenges are not stored aboard the device. Hence capturing and/or hacking into a device will not reveal its secret identity, i.e., its fingerprint. However, it has been shown that advanced ML techniques could successively model the PUF operation using some intercepted CRPs for training without even knowing the process variation details [4]. PIDA mitigates vulnerability to modeling attacks in presence of individual and colluding actors. Finally, a trusted server is assumed to enroll nodes, prior to their participation in the network. Yet, such a server is not engaged in coordinating the IoT operation and is not consistently reachable to the nodes.

B. Attack Model

PIDA opts to counter attempts to gain access to an IoT network by impersonating legitimate nodes. Upon joining the network, the adversary can launch attacks such as false data injection and selective forwarding, message relay, etc. The adversary pursues two strategies to uncover the security provisions employed by the individual nodes, i.e., modeling the embedded PUFs. The first is to eavesdrop on the communication links of each device (prover) to intercept authentication messages and collect CRPs for developing a model that replicates the functionality of the device's PUF. The second strategy is to hack (intrude) into the device to uncover the stored CRPs. For PIDA, the adversary may target multiple verifiers to collect CRPs for a certain prover and consolidate the gained knowledge. As will be explained, PIDA thwarts such a threat by providing verifier-specific responses using a one-way function, which avoids the communication of any information that is directly related to the PUF. Note that impersonation of the verifier does not present any relevance as the objective is to authenticate the provers.

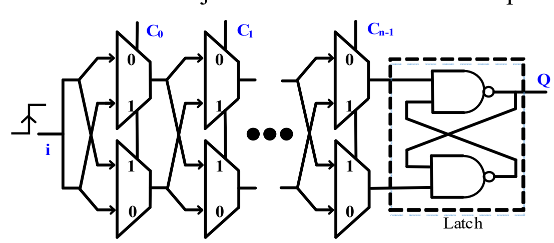

Fig. 1. Schematic diagram of an Arbiter PUF, where the challenge bits control the individual multiplexers and cause the input signal to experience different delays on distinct devices and consequently the latched value (Q) would differ.

C. Approach Overview

PIDA enables peer-to-peer device authentication without using a trusted authority. The main idea is to employ a lightweight hardware primitive, namely a PUF, where the node's identity can be verified using the PUF response to a given challenge bit-stream. A key advantage of the PUF is that the response does not have to be stored in the prover's memory and hence the device becomes resistant to tampering and intrusion attempts. However, in distributed setups, the verifier needs to know some CRPs for the prover's PUF to validate the response. Sharing CRPs raises a concern about the potential of leakage that allows an adversary to model the PUF and impersonate the prover [4]. One option for addressing such a concern is to pursue a decentralized authentication process and limit the number of shared CRPs so that if a verifier node is captured or hacked, the leaked CRPs do not suffice for developing an accurate model of the PUF. This approach, however, has two main disadvantages. First the limited CRPs makes the system susceptible to replay attacks since challenges need to be frequently repeated. Second, if multiple verifier nodes are captured or hacked, the uncovered CRPs could be aggregated to model the prover's PUF, a scenario that is being viewed as a collusive attack.

To overcome the two aforementioned disadvantages, PIDA employs two main features: (i) the response to a given challenge is made to be a function of the verifier's ID, and (ii) the response is obfuscated such that an adversary cannot unmask the effect of the verifier ID and infer the actual PUF output. In other words, the verifier holds only transformed responses that can validate a prover's identity without tabulating the prover's PUF output. Fig. 2 illustrates the PIDA protocol. To authenticate a prover, the verifier picks a challenge, C, that has the response $\Re$ for. The prover will apply the provided C as an input to its PUF and note the output R_C. The prover also will use the verifier ID as an input to the PUF and get R_{ID}. To mitigate the effect of noise on the PUF output, the verifier also provides ECC for both

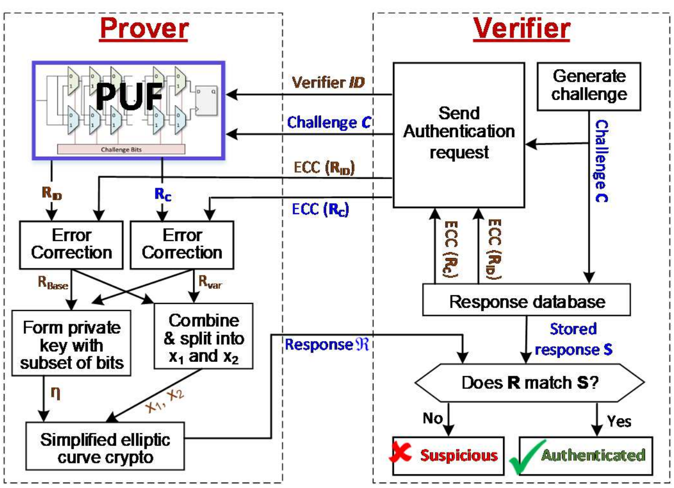

Fig. 2. An overview of the operation of PIDA. The verifier sends to the prover an authentication request that includes a challenge bit stream, C. The prover uses the PUF output for C and the output when using the verifier ID as PUF input, in forming a response to the request. Error correction codes are also sent along the request to help the prover mitigate the effect of noise on the PUF output. The request response will be generated by feeding a transformed variant of the PUF outputs to a simple Elliptic Curve function whose parameters are known only to the prover. The response will be matched by the verifier to a pre-known value that it obtains at the time of enrollment in the system.

R_C and R_{ID}. The verifier gets C, $\Re$, ECC(R_C), and ECC(R_{ID}) at the time of device enrollment; yet the verifier will not know R_C and R_{ID} themselves. The prover will use the error correction code to fix any bit flips in the PUF output within R_C and R_{ID} due to noise, and generate R_{var} and R_{base}, respectively. We note that R_{base} does not change for requests made by the same verifier.

To generate $\Re$, R_{var} and R_{base} are then used to form input points to a simple ECF whose parameters are specific and known only by the prover. The details of such a process will be provided in the next section. The verifier will match $\Re$ to the value it has to validate the authenticity of the prover. We note that the value of $\Re$ is real rather than Boolean. PIDA is lightweight in terms of the processing overhead. Due to the use of ECF, the responses provided to the same verifier cannot be correlated to infer R_C from $\Re$. More importantly, by providing an ID-based response, PIDA ensures that the response for the same challenge differs among the various verifiers and thus thwarts threats of collusion and authentication message replay. In the next section, we will discuss PIDA design in detail.

IV. Distributed PUF-Based Device Authentication

A. Detailed Protocol Steps

PIDA consists of the following two phases, reflecting the times for sharing security parameters and using them, respectively.

Initialization and Enrollment Phase: When a device D_p joins the network, it needs to be informed about how to authenticate other nodes, and also share information for how itself could be authenticated. PIDA assumes that a trusted server is involved only in such a phase. The server's role is to act as a database of the IDs of active (enrolled) devices in the network. The server can also verify the authenticity of devices prior to enrolling them; such a process may be based on manufacturing settings, and is outside the scope of PIDA. Otherwise the server has no information about the security primitives of the enrolled devices. Meanwhile, a PUF is embedded during the design of each device. The device will be initialized prior to deployment. During initialization, a device, D_p, will be exclusively provided with the following parameters that no other node knows about:

α_p and β_p: are the coefficients for the ECF used by D_p to generate the response (device secret), which can be viewed as an obfuscation of the involved outputs of PUF_p. The values of α_p and β_p are picked by the node and would thus vary per node; they also are not known to any verifier D_v.

max_p: is a bound on the ECF range and is referred to as the key size in the realm of elliptic curve cryptography. The setting of max_p will be discussed later in this section.

$|\eta|$: is the size of the private key used by D_p and is less than m.

We note that the length of a node identifier should not exceed the size of the PUF so that the ID of a verifier D_v can be applied to PUF_p. For example, if the ID length is l bits, a fixed pattern for the most significant 2^{n-l} bits could be assumed by D_p before applying to its PUF.. Thus, based on the aforementioned parameters, D_p will be able to generate $\Re$ for each $(C, R_C) \in \Psi_p$, where C corresponds to a verifier ID and Ψ_p is the set of CRPs for PUF_p. Let ECC_p be the set of error correction codes for the responses in Ψ_p:, i.e., $ECC_p = \{ECC(R_C) \mid (C, R_C) \in \Psi_p\}$.

When D_p contacts the server to be enrolled, it in turn will introduce D_p to the network. D_p will then receive $\Phi_{w\rightarrow p}$ from each enrolled device D_w. $\Phi_{w\rightarrow p}$ is a set of challenge and obfuscated responses of node D_w when queried by D_p, where $\Phi_{w\to p}$={$(C, \Re)$ | $\Re=ECF_w(R_{ID_p}, R_C)$ & $(C, R_C)\in \Psi_w$, $(p, R_{ID_q})\in \Psi_w$}. We note that only a subset of Ψ_w is shared with D_p, i.e., $|\Phi_{w\to p}| << |\Psi_w|$. We stress that D_p will never get the PUF output of any other node D_w in the system; only the obfuscated responses are shared. Similarly, D_p will form the set $\Phi_{p\to w}$ for each D_w in the network. Obviously, such an approach requires reaching D_w; hence the enrollment phase could be staggered based when D_p and D_w become in range of one another for the first time. PIDA supports dynamic enrollment and response regeneration on the fly by the prover. The shared responses $\Re$ are verifier-specific which ensures scalability and prevents collusion. After enrollment, the server's role seizes.

Authentication Phase: When a node D_q wants to establish a communication link with D_p, e.g., to receive data, D_q first needs to confirm the identity of D_p. In such a case D_q acts as a verifier and sends an authentication request to the prover, D_p. To do so under PIDA, D_q will randomly pick an entry $(C, \Re)$ from the set $\Phi_{p\to q}$ and include both C and ECC(R_C) in the authentication request. The request packet implicitly has the ID of D_q. In addition, the request includes $\text{ECC}(R_{ID_q})$ since PIDA factors in the verifier ID. In turn, the prover feeds C and the verifier's ID into its PUF to obtain R_C and R_{ID_q}, respectively. To ensure the integrity of the PUF output, the ECC provided by the verifier is used to correct any bit flip. The selection of the ECC generation algorithm is beyond the scope of this paper; yet the chosen algorithm should yield an ECC that is not intertwined with the actual bit string so that it can be provided separately from the actual response. As we explain, storing the ECC of the prover's PUF response at the verifier does not constitute much leakage since the verifier only has the obfuscated response $\Re$ rather than the PUF output and one cannot infer R_C from $\Re$.

The prover, D_p, uses the corrected version of R_C and R_{ID_q}, denoted as R_{base} and R_{var}, respectively to generate the response to the verifier. The idea is to mix the bits of R_{base} and R_{var}, and then split the formed bit string into three parts: $|\eta|$, X_1 and X_2. These three parts are used to generate a unique response through a one-way function, namely, a simple ECF, where $|\eta|$ reflects the private key size and X_1 and X_2 are the x-coordinates for two points on the elliptic curve. The idea, which is inspired by the elliptic curve cryptography, is to define a line using two points on the curve and find another point where such a line intersects again with the curve. Such a process is repeated η times, where the x-coordinate of the last found point is used as a response to the authentication request. Such a process is explained in detail in the next subsection. By not knowing how R_{base} and R_{var} are combined, how the formed bit string is divided into $|\eta|$, X_1 and X_2, and what ECF is employed, it is not possible for an attacker that intercepts the authentication packets or even hack D_q, to correlate R_C to the corresponding $\Re_C$ and model the prover's response generation process. Moreover, $\Re_C$ is generated while factoring in the verifier ID, and is not similar across verifiers; hence a collusive aggregation of $(C, \Re_C)$ across multiple verifiers will be ineffective for building an accurate ML model. Also, a response replay will be invalid across distinct verifiers.

B. *Response Obfuscation*

A PUF constitutes a lightweight mechanism for generating authentication tokens. As pointed out earlier, obfuscating the PUF output or limiting the number of shared CRPs are the conventional means to mitigate PUF modeling vulnerabilities. However, constraining the count of shared CRPs does not scale for dynamic networks where colluding actors may succeed in modeling the PUF by aggregating CRPs for the same prover across multiple verifiers. Meanwhile, if a prover D_p applies the same obscuration method for all verifiers, hacking one verifier, D_q or intercepting its communication with D_p, will allow the adversary to replay D_p's response when another verifier uses the same challenge bit-stream and consequently impersonate D_p.

PIDA addresses the aforementioned issues by pursuing an identity-based response generation per challenge. To mitigate the risk of leaked CRPs, PIDA avoids direct usage of CRPs and instead establishes a security association between a prover and a verifier that hides the correlation between the challenge and PUF response. Such association is irreversible and can be realized using a one-way function. To mitigate the complexity of the response generation, PIDA leverages the properties of elliptic curves where reverse-engineering of a response is extremely difficult, if not infeasible, where there is not any polynomial-time algorithm for doing so [20]. PIDA, also makes the private key unknown and variable, which further increases resilience to attacks. Basically, a prover defines an ECF in the form $y^2 = x^3 + \alpha x + \beta$, referred to as Weierstrass form, where the discriminant $(4\alpha^3 + 27\beta^2) \neq 0$ so that the cubic does not have a repeated root. According to the properties of ECF, a line between any two points on the curve will intersect the curve at exactly one more than one additional point. Let $L_1 = (x_1, y_1)$, and $L_2 = (x_2, y_2)$ are two points on the curve, The line $\overline{L_1L_2}$ intersects with the curve at $L_3 = (x_3, y_3)$, where:

$$s = \begin{cases} \dfrac{y_1 - y_2}{x_1 - x_2} & if\ x_1 \neq x_2, \\ \dfrac{3{x_1}^2 + \alpha}{2y_1} & if\ x_1 = x_2 \end{cases}$$
$$x_3 = (s^2 - x_1 - x_2) mod\ max_p \qquad (1)$$
$$y_3 = (-s(x_3 - x_1) - y_1) mod\ max_p$$

In Elliptic curve cryptography, finding the new point is called the dot operator. To further apply the dot operator, the new (third) point is to be reflected over the x-axis and the intersection point with the curve is connected to L_1 to define the next point. The number of consecutive dot operations is referred to as the private key, η. Fig. 3 shows an example, where the line $\overline{L_1L_2}$ intersects with the curve at L_3. When reflecting L_3 over the x-axis and connecting the intersection point with L_1, a new point L_4 is defined. When reflecting L_4 again and connecting with L_1, a new intersection point L_5 is found, and so on. For this example, three dot operations are applied and hence η=3. Since the intersection points can be anywhere on the curve, a bound is defined (max_p in Fig. 3) to limit the range of values, which constitute the range of $\Re$ values in PIDA. Using modulo operation simply forces a wraparound so that the line intersects with the curve from the other side of the y-axis. The modulo operation can also discretize the dot operation by theoretically enumerating possible intersection points into a finite set Ω, for which the result of a dot operation of two points in Ω is a third point in Ω; such discretization would simplify calculating the new point. Setting max_p to be a prime number is shown to increase robustness [20]; yet in PIDA max_p is set to 2^θ, where $m < \theta \leq 2m$. The rationale is that using prime numbers elevates the complexity of modulo operation, which is undesirable for IoT and is not warranted in PIDA given the many other features that ensure robustness against attacks. A large value of θ is desirable to increase the range and make the output of the ECF to be more unpredictable. Yet, the value of θ varies among devices, which adds to the strength of PIDA. The following summarize how PIDA generates the obfuscated response $\Re$ using R_{base} and R_{var}:

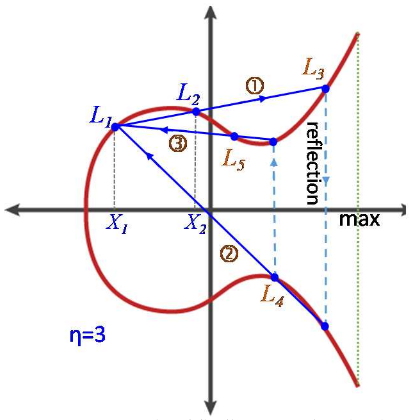

Fig. 3: An example of finding an authentication response for η=3. The points L_1 and L_2 are determined on the elliptic curve using the values X_1 and X_2 derived from the combined (and corrected) PUF output. The max line provides a bound (key size) that forces a wraparound through modulo operation.

1) The bits of R_{base} and R_{var} are mixed. Such a mix can be prover dependent. To simplify the discussion, let us assume that the bits are alternated, meaning that every consecutive two bits in the combined bit strings do not belong to either R_{base} or R_{var}. For example, if $R_{base} = \{d_0, d_1, ..., d_{m-1}\}$ and $R_{var} = \{e_0, e_1, ..., e_{m-1}\}$, the combined string would be, $\xi = \{e_0, d_0, e_1, d_1, ..., e_{m-1}, d_{m-1}\}$.
2) Noting that $|\xi|=2m$ bits, a subset of $|\eta|$ bits is picked, where $|\eta|$ is an even number. Such a subset is used to define the value of η for the response of the authentication request. Consequently, the value of private key varies per challenge. For example, if $|\eta|=2$, two bits at the beginning, middle or end of ξ could be extracted to determine η.
3) The remaining $2m$-$|\eta|$ bits in ξ are then divided into two equal-sized sets of bits, that reflects X_1 and X_2.
4) X_1 and X_2 are used to determine L_1 and L_2 on the ECF. We note that the ECF parameters differ among nodes, and hence the ECF is prover-specific.
5) The dot operation is applied on the curve η times as explained above. The x-coordinate of the final point is used as $\Re$. The value of *max* will constrain the range of $\Re$.

V. Performance Evaluation

To demonstrate the robustness of PIDA against individual and colluding modeling attempts, we have implemented an arbiter-PUF on Xilinx ARTIX-7 FPGA. The PUF is used to map a 64-bit challenge to 64-bit output. In the experiments we considered one prover, and four verifiers, each of which is given a subset of the prover's challenges and their corresponding obfuscated responses according to PIDA, and the ECC of the actual PUF response. i.e., the set $\Phi_{w \to p}$ according to our used notation. A Neural Network (NN) is used as the underlying attack technique to model prover's operation, i.e., PIDA. The employed NN is a regression model with 5-layer fully connected architecture with one input layer (with neuron count reflecting the PUF size), three nonlinear hidden layers (with 5, 10 and 15 neurons) and one output neuron. Rectified linear unit (ReLU) is used as an activation function in all layers. The learning rate and momentum are 0.01 and 0.99, respectively, and the number of epochs is 1000. In PIDA, the ECF is picked by the prover and hence may vary across the network. In the experiment we have set $\alpha = -10$, and $\beta = 26000$ so that the ECF has a single root at -29.74, which ensures continuity along the x-axis.

Two attack scenarios are considered: (i) a single verifier, and (ii) multiple collusive verifiers; in both cases the attack is geared for modeling the authentication process to predict the prover's response for unknown challenges. We assess the performance in terms of the modeling accuracy and the energy overhead. We compare the performance to the baseline case where only the PUF response is used, and to two variants of PIDA. The two variants reflect discretization of the ECF dot operation, which reduces the computational complexity. In the first variant (PIDA-DF1) all $2m$-$|\eta|$ bits are used to determine one point that is dotted by itself $|\eta|$ times. This reflects a higher precision in determining the point coordinates on the curve. The second variant, PIDA-DF2, uses two points, similar to PIDA, with discretized dot operation (see Section IV-B). We study the effect of the following parameters: (i) max_p (2^θ), which controls the range of values that PIDA uses as response, (ii) $|\eta|$, which reflects the number of dot operations, and the number of colluding attackers (compromised nodes).

Fig. 4 compares PIDA to a PUF only based authentication mechanism in terms of resilience to modeling attacks. As seen in the figure, intercepting enough CRPs for the PUF enables an adversary to devise an accurate model. This is not the case when using PIDA, where the model accuracy is so low that the prediction reflects no more than a random guess. This is very much attributed to the real, rather than Boolean, response that PIDA provides. The figure demonstrates the increased robustness that PIDA achieves, even when many CRPs are used. Tables 1-3 compare the performance of PIDA, to its two variants, namely, PIDA-DF1 and PIDA-DF2. Table 1 studies the effect of max_p, which determines the range of x values for the curve points (used by PIDA as response). The results show clearly that the discretization does not guarantee response unpredictability, particularly for small max_p values. Meanwhile, PIDA sustains robustness due to the high precision of the response and the adversary's inability to successfully guess the range of possible values. The discretization vulnerabilities lie in the fact that max_p is used in Eq. (1) in the modulo operation. We note that the use of a single point is generally better except when max_p is very small, e.g., θ=2, since very few values are possible.

Clearly the results in Table 2 demonstrate that PIDA's robustness is not impacted much by the private key size, even

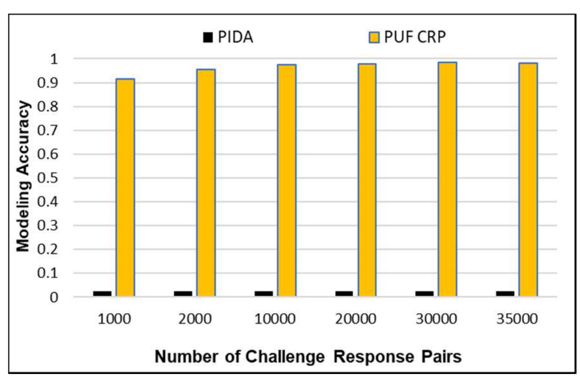

Fig. 4: The modeling accuracy for PIDA versus the use of PUF alone. PIDA provides a floating point response, which is not possible to predict correctly.

for PIDA-DF1 and PIDA-DF2. This is attributed to the randomness imposed by the PUF in selecting the initial points and the number of dot operations. The most relevant observation here is that a small |η| would suffice from a security point of view, which enables PIDA to be lightweight since the number of dot operations will be reduced, as will be shown. While Tables 1 and 2 consider a single attacker, Table 3 assesses the modeling accuracy when multiple malicious verifiers collude. As indicated by the results, PIDA nullifies the advantage of any collaboration among attackers. We can see that even PIDA-DF1 and PIDA-DF2 are not impacted by collusion as the response generation factors in the verifier ID instead of being generic for any communicating nodes. Finally, Table 4 compares the additional overhead for PIDA, and its first (best) variant relative to a baseline case where the PUF's CRPs are exchanged in plaintext. The additional processing is due to forming ξ and applying the dot operation of ECF $2^{|\eta|-1}$ times. Table 4 reports the extra energy needed to conduct one round of authentication (total for communication and communication) based on the Digi XBee 3 Zigbee radios with an energy per bit 360 *nJ/bit* and an Arduino platform with an active current of 1.23 mA when clocked at 16 MHz. The results clearly favor the usage of a small key size, |η|. The overhead for a large |η| is significant in case of PIDA, which is attributed to the increased (exponential growth) floating point operations; yet such a key size is unwarranted. Considering all results collectively, one concludes that using |η|=2, and θ ≥ 8 is the best configuration of PIDA in terms of security and energy overhead.

VI. Conclusions

This paper has presented PIDA, a novel lightweight authentication scheme for IoT. PIDA leverages the advantages of PUFs in terms of tamper-resistance and low overhead. Unlike existing PUF-based protocols, PIDA supports distributed operation where no centralized server is engaged in mutual authentication of device pairs. To effectively realize distributed authentication, PIDA factors in the device ID in obfuscating the response of the PUF and employs Elliptic Curve functions to guard against ML modeling attacks conducted by a single or multiple collusive nodes. The validation results have shown that PIDA is lightweight and very robust against modeling attacks, where an adversary cannot achieve meaningful accuracy in predicting the prover's response to any challenge.

Reference

[1] T. A. Ahanger and A. Aljumah, "Internet of things: A comprehensive study of security issues and defense mechanisms," *IEEE Access*, vol. 7, pp. 11 020–11 028, 2019.

[2] F. H. Al-Naji, and R. Zagrouba, "A survey on continuous authentication methods in Internet of Things environment," *Computer Communications*, vol. 163, pp. 109-133, 2020.

[3] A. Shamsoshoara et al., "A survey on physical unclonable function (puf)-based security solutions for internet of things," *Computer Networks*, vol. 183, p. 107593, 2020.

[4] M. Khalafalla and C. Gebotys, "PUFs deep attacks: Enhanced modeling attacks using deep learning techniques to break the security of double arbiter PUFs," *Proc. of Design, Automation and Test in Europe (DATE 2019)*, pp. 204–209, Florence, Italy, Mar 2019.

[5] Y. Atwady, and M. Hammoudeh. "A survey on authentication techniques for the Internet of Things," *Proc. Int'l Conf. on Future Networks and Distributed Systems (ICFNDS '17)*, Cambridge, UK, Jun 2017, p. 8.

[6] G. Dessouky, T. Abera, A. Ibrahim, and A.-R. Sadeghi, "LiteHAX: lightweight hardware-assisted attestation of program execution," *Proc. of IEEE/ACM Int'l Conf. Comp.-Aided Des. (ICCAD)*, San Diego, CA, 2018.

[7] F. H. Al-Naji, and R. Zagrouba, "A survey on continuous authentication methods in Internet of Things environment," *Computer Communications*, vol. 163, 2020, pp. 109-133.

[8] K. Lounis and M. Zulkernine, "Lessons Learned: Analysis of PUF-based Authentication Protocols for IoT," *ACM Digital Threats: Research and Practice,* September 2021, https://dl.acm.org/doi/10.1145/3487060.

[9] S. S. Zalivaka et al., "Reliable and modeling attack resistant authentication of arbiter PUF in FPGA implementation with trinary quadruple response," *IEEE Trans. on Info. Forensics & Sec.*, vol. 14, no. 4, pp. 1109–1123, 2019.

[10] M. A. Qureshi and A. Munir, "puf-rake: A PUF-based robust and lightweight authentication and key establishment protocol," *IEEE Trans. on Dependable and Secure Computing*, pp. 1–1, 2021.

[11] M. Ebrahimabadi, M. Younis, W. Lalouani, and N. Karimi, "An Attack Resilient PUF-based Authentication Mechanism for Distributed System," *Proc. the VLSI Design Conference (VLSID 2022)*, Feb. 2022.

[12] M. Ebrahimabadi, M. Younis, W. Lalouani, and N. Karimi, "A Novel Modeling-Attack Resilient Arbiter-PUF Design," *Proc. the VLSI Design Conference (VLSID 2021)*, Feb. 2021.

[13] C. Gu et al., "A modeling attack resistant deception technique for securing PUF based authentication," *Proc. AsianHOST*, 2019, pp. 1–6.

[14] F. Farha et al., "SRAM-PUF based entities authentication scheme for resource-constrained iot devices," *IEEE Internet of Things Journal*, vol. 8, no. 7, pp. 5904-5913, April 2020.

[15] M. Aman, M. H. Basheer and B. Sikdar, "Data Provenance for IoT with Light Weight Authentication and Privacy Preservation," *IEEE Internet of Things Journal*, Vol. 6, No. 6, pp. 10441-10457, Dec. 2019.

[16] Q. Jiang, et al., "Three-factor authentication protocol using physical unclonable function for IoV," *Comp. Comm.*, Vol. 173, 2021, pp. 45-55.

[17] M. Barbareschi et al., "A PUF-based mutual authentication scheme for cloud-edges iot systems," *Future Generation Computer Systems*, vol. 101, pp. 246–261, 2019.

[18] Y. Nozaki and M. Yoshikawa, "Secret Sharing Scheme Based Secure Authentication for Physical Unclonable Function," *Proc. of the IEEE 4th Int'l Conf. on Computer and Communication Systems*, pp. 445-449, 2019.

[19] K. Lounis and M. Zulkernine, "T2T-MAP: A PUF-Based Thing-to-Thing Mutual Authentication Protocol for IoT," *IEEE Access*, vol. 9, pp. 137384-137405, 2021.

[20] M. Robshaw, "Trapdoor One-Way Function," in van Tilborg H.C.A. (eds) Encyclopedia of Cryptography and Security. Springer, Boston, 2005.

Table 1: Effect of max_p (2^{θ}) on the modeling accuracy with |η|=32.

	Modeling accuracy				
	θ=2	θ=4	θ=5	θ=7	θ=8
PIDA	≈0	≈0	≈0	≈0	≈0
PIDA-DF1	54%	6%	2%	0.2%	≈0
PIDA-DF2	28%	10%	7%	1.1%	0.5%

Table 2: The impact of |η| on the modeling accuracy when **θ=7**.

	Modeling accuracy				
	|η|=2	|η|=4	|η|=8	|η|=16	|η|=32
PIDA	≈0	≈0	≈0	≈0	≈0
PIDA-DF1	0.5%	0.1%	0.1%	0.1%	0.1%
PIDA-DF2	3%	1.5%	0.4%	0.2%	0.1%

Table 3: Effect of collusion on modeling accuracy (|η|=32and **θ=2**).

	Modeling accuracy			
#Attacker	1	2	3	4
PIDA	≈0	≈0	≈0	≈0
PIDA-DF1	54%	≈0	≈0	≈0
PIDA-DF2	28%	≈0	≈0	≈0

Table 4: Response obfuscation overhead as a function |η| and max_p.

	Energy Overhead (nJ)			
	|η|=2	|η|=4	|η|=8	|η|=16
PIDA (θ=5)	0.32	0.38	2.6	896.04
PIDA (θ=7)	0.29	0.34	0.73	1607.77
PIDA-DF1(θ=5)	0.26	0.32	0.38	0.38
PIDA-DF1(θ=7)	0.28	0.30	0.34	0.38