IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017. Doi Number

A Lightweight PUF-Based Authentication Protocol Using Secret Pattern Recognition for Constrained IoT Devices

Tarek A. Idriss[1], Member, IEEE, Haytham A. Idriss[2], and Magdy A. Bayoumi[2], Fellow, IEEE
[1]Department of Computer Science, Western Washington University, Bellingham, WA 98225 USA
[2]Center for Advanced Computer Studies, University of Louisiana at Lafayette, Lafayette, LA 70503 USA

Corresponding author: Tarek A. Idriss (e-mail: Tarek.Idriss@wwu.edu).

ABSTRACT PUFs, or physical unclonable functions, are hardware security primitives that can offer lightweight security solutions for constrained devices through challenge-response authentication protocols. However, the lightweight PUF-based security solutions that have been presented often lack security features such as mutual authentication or message encryption, which could be vital for many applications. Other protocols suffer from vulnerabilities to denial of service attacks that make them impractical to use. This work introduces a lightweight PUF-based protocol that uses secret pattern recognition to offer mutual authentication and authenticated secret message exchange for constrained devices on the Internet of Things. The protocol utilizes several techniques to introduce nonlinearity, and it can employ any strong PUF circuit for which a soft model can be generated. The authentication process requires simple bitwise operations along with a PUF circuit and a true random number generator (TRNG). By avoiding the use of any cryptographic or hash functions, the protocol's lightweight nature is preserved. The security of the proposed protocol against modeling attacks is tested to showcase its resilience. Similar PUF-based protocols are investigated and found to lack some essential security features.

INDEX TERMS authentication, hardware security, IoT security, lightweight security, physical unclonable functions.

I. INTRODUCTION

The establishment of reliable security for constrained devices has been an ongoing challenge due to the high constraints on power consumption, implementation area, and device cost. Standard cryptographic solutions that provide provable security have prohibitive area and power demands for many applications, such as radio frequency identification (RFID) tags, medical implants, or smart cards. For instance, low-cost RFID tags can only use 3-5K logic gates for security functions, as reported in [1], [2], while public cryptography algorithm implementations, which are crucial for reliable key exchange [3], [4], can use between 12K and 22K logic gates [5]. Silicon-based physical unclonable functions (PUFs) [6] are emerging hardware security primitives that have the potential to offer security solutions for constrained devices due to their small implementation overhead.

PUFs have already established a foothold in various IoT applications [7], [8], as low power security has been in great demand. PUFs exploit the inherent randomness within complementary metal–oxide–semiconductor (CMOS) devices' manufacturing process to produce a unique response when offered an input challenge. As the PUF response depends on the randomness of the minuscule variations within the manufacturing process, PUFs are inherently unique and unclonable. The challenge-response space of a PUF determines whether it is classified as a weak PUF or a strong PUF [9]. Weak PUFs are characterized by a relatively small challenge response. Such PUFs are primarily used as alternatives to traditional key storage. Strong PUFs, on the other hand, have ample challenge-response spaces, making them more useful for challenge-response authentication protocols.

Although unclonable through hardware, strong PUFs are vulnerable to modeling attacks [10]–[13]. In such attacks, adversaries collect the exchanged challenge-response pairs (CRPs) used in authentication sessions and apply machine learning algorithms to produce a soft model of the PUF circuit. These soft models are capable of correctly predicting the response of any incoming PUF challenge. Hence, PUF-based authentication protocols must either employ very large PUF circuits to ensure resilience against modeling attacks or use expensive computations that include encryption or hashing, as suggested in [14]–[26]. Both approaches could introduce significant implementation overhead, making these protocols prohibitive for use in constrained devices. As an alternative, PUF-based authentication protocols that use simple pseudocryptographic algorithms were suggested in [27]–[32]. Such protocols can offer secure device authentication while exhibiting high resilience against modeling attacks. However, most of these protocols lack essential security features such as mutual authentication or resistance against denial of service (DoS) attacks that could render a device completely useless. Furthermore, none of the introduced lightweight protocols offer secure authenticated secret message exchange. The lack of a secure and lightweight PUF-based protocol that can offer an unlimited number of mutual authentications and secret message exchanges is the motivation behind this work.

We introduce a lightweight PUF-based authentication protocol that can offer an unlimited number of mutual authentications and secret message exchanges to constrained devices on the IoT. A method for securing the protocol against man-in-the-middle (MITM) attacks is incorporated in the protocol design. The lightweight PUF-based authentication (LPA) protocol establishes its high resilience against modeling attacks by assigning a set of hidden exchange patterns that are unique to each device. The protocol's resilience against machine learning attacks is showcased by testing it against known machine learning-based attacks such as evolution strategies (ES), artificial neural networks (ANNs), and support vector machines (SVMs). We list the main contributions of this work:

- We introduce a lightweight PUF-based mutual authentication protocol that recognizes the secret patterns assigned to devices.
- We introduce a method for authenticated secret message exchange that offers message encryption and guarantees both the secrecy and origin of each message.
- We introduce novel challenge transformation functions that can transform the highly correlated challenges of arbiter-based PUFs into multiple unique uncorrelated challenges while requiring a small implementation overhead. The functions are then utilized to protect the protocol against MITM attacks.
- We present a security analysis of the protocol and compare its security features with other recently introduced lightweight PUF-based authentication protocols.

The remainder of this paper is organized as follows. Section II is a background on PUFs and PUF-based authentication protocols. Section III introduces the challenge-challenge exchange concept and the LPA protocol. Section IV introduces the uncorrelated challenge transformation functions that can be used on arbiter-based PUF challenges. Section V presents a detailed security analysis of the protocol and tests the protocol's resilience against known machine learning attacks. In Section VI, the error tolerance of the protocol is illustrated. In Section VII, a delay and throughput analysis of the protocol is presented. The features of the introduced protocol are compared with those of the methods introduced in related work in Section VIII. Section IX concludes this paper.

II. PUFS IN AUTHENTICATION PROTOCOLS

A. DELAY ARBITER PUF

We examine the delay arbiter PUF [6], as it is one of the earliest and most studied silicon-based PUFs. The arbiter PUF also serves as a building block in other more complex, strong PUF designs. The arbiter PUF compares the delays of two identical paths to generate either a '0' bit or a '1' bit. Although the two paths are identical and should introduce the same delay, unpredictable minuscule variations during the fabrication process ensure that one path is ultimately faster than the other. Multiplexers, referred to as 'switch components,' are inserted into the paths. Challenge bits are used as the selected inputs of the multiplexers. Each switching component introduces either crossed paths or straight paths depending on the multiplexer's selected bit, as shown in Fig. 1. This results in an exponentially large number of possible paths.

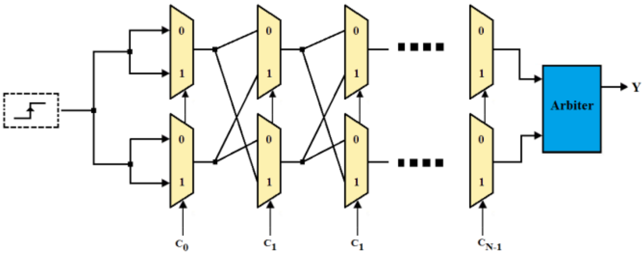

FIGURE 1. **Delay arbiter PUF architecture.**

B. MODELING ATTACKS ON ARBITER PUFS

The arbiter PUF was found to be vulnerable to modeling attacks in [10]–[12]. In such attacks, an adversary collects the exchanged CRPs used in the authentication sessions and applies machine learning algorithms to produce a software model of the PUF. This soft model is capable of correctly predicting the responses to new challenges. The arbiter PUF can be modeled as a set of delay elements. The delay difference Δ at the arbiter can be expressed as a function of the differential delay vector ω and Φ, the feature vector that is a function of the input challenge [10]:

$$\Delta = \omega^T \Phi \tag{1}$$

Various machine learning algorithms can be used to determine the separating hyperplane $\omega^T \Phi = 0$ that serves as the decision boundary surface for the response bit. Linear regression (LR) has been shown to be a very efficient algorithm in terms of solving for ω. For a 64-stage arbiter PUF, observing 640 CRPs would allow an adversary to produce a soft model of the PUF with 95% accuracy and a short training time (< 1 sec). Several modifications of the arbiter PUF have been suggested to enhance its security. The XOR-Arbiter PUF [33] combines several rows of arbiter PUFs into a single bit. Other enhanced designs, such as the lightweight secure PUF (LS-PUF) [34], have been suggested to increase the resistance of PUFs to modeling attacks. However, these enhanced designs were also found to be vulnerable, albeit to a lesser extent, to modeling attacks in [10]. For highly nonlinear PUFs, such as the feedforward arbiter PUF (FF-PUF) [35], [36], machine learning techniques utilizing evolution strategies (ES) have been utilized to produce soft models of the PUFs. It was suggested in [10] that modeling-resilient PUF designs could be possibly implemented by drastically increasing the number of XOR-ed PUF circuits. However, such PUFs have been shown in [37] to require a large implementation area, making them infeasible to implement in constrained devices.

C. PUFS IN AUTHENTICATION PROTOCOLS

Fig. 2 shows a ‘bare-bones’ PUF-based authentication protocol. A server with access to a soft model of the PUF could generate a set of challenges C and send it to the device. The device could then use its PUF circuit to generate a set of responses R for the challenges. These responses are sent back to the server. The server compares the device’s responses with those generated from the soft model. If the received and generated responses match within a certain margin α, the device is deemed authentic.

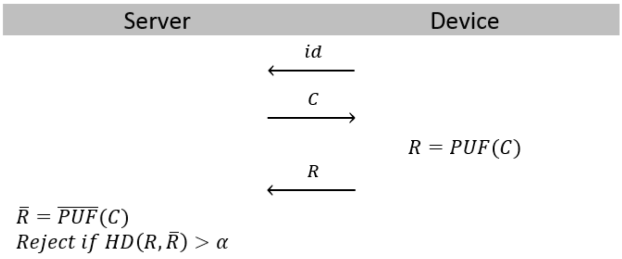

FIGURE 2. A ‘bare-bones’ PUF-based challenge-response authentication protocol.

The authentication protocol in Fig. 2 is insecure, as an adversary can perform a modeling attack by collecting the exposed CRPs. PUF-based authentication protocols use varying methods to obscure the correlations between the challenges and responses. The most popular approaches utilize cryptographic or hash functions for hiding the correlations. Such schemes have been suggested in [14]–[26] for various applications. Cryptographic or hash functions provide reliable security but at the cost of using expensive computations that might not suit many constrained devices.

To satisfy constrained devices’ security needs, lightweight PUF-based authentication protocols that utilize simple pseudocryptographic algorithms and avoid hash functions have been suggested [27]–[32]. However, the lightweight solutions suggested so far lack essential security features, which often limits their usability. A more detailed review of lightweight protocols is presented in Section VII. The introduced LPA protocol aims to offer a complete lightweight security solution for constrained devices by offering features such as mutual authentication and secret message exchange.

III. THE LPA PROTOCOL

A. PROTOCOL SETUP AND ADVERSARY MODEL

The proposed protocol utilizes a strong PUF circuit at the constrained device side and a soft PUF model at the unconstrained device side. The soft model can be obtained by performing a machine learning attack on the raw challenge-response pairs of the PUF through access to special measurement points. These access points are then permanently disabled before deployment. The majority of PUF-based protocols employ this approach. The strong PUF circuit is treated as a black box by the protocol, and hence, any strong PUF circuit can be used as long as it has an associated soft PUF model that can be shared with trusted parties. We note that weak PUF circuits would not be suitable because they have small challenge-response spaces.

Fig. 3 shows an illustration of how the protocol can be deployed on the device side and server side. On the server side, the protocol is implemented via software. When a device cannot be trusted with permanent access, the PUF model and the protocol logic can be stored on a remote trusted server, while devices with temporary access can forward authentication requests to the trusted remote server. This remote deployment allows system administrators to remove devices’ access rights by revoking access to the remote model and logic.

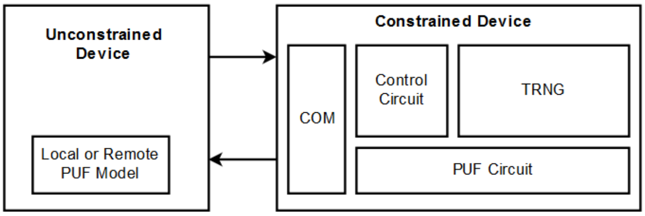

FIGURE 3. Overall LPA protocol architecture.

The adversary is assumed to have access to the communication channel used by the prover and verifier. The adversary can intercept the communicated messages and may also perform MITM substitution attacks. Protecting against probing attacks or side-channel attacks is outside the scope of this work. A description of the annotations used by the protocol is provided in Table I.

TABLE I
DESCRIPTION OF ANNOTATIONS

Symbol	Description
Cj	A pseudochallenge used to generate an input challenge by applying a transformation function to it. $C1$ is the verifier’s pseudochallenge, while $C2$ and $C3$ are the prover’s pseudochallenges.

Ti	A transformation function that transforms a single pseudochallenge Cj into a transformed input challenge $Ti(Cj)$.
$PUF\big(Ti(Cj)\big)$	The single-bit output/response of the PUF circuit when presented with a transformed challenge $Ti(Cj)$. Only transformed challenges are fed to the PUF circuit.
R_{Vi}	The PUF response bit of the transformed verification challenge $Ti(C1)$.
R_V	The concatenation of three response bits R_{V1}, R_{V2} and R_{V3}. In every exchange, three transformations $T1$, $T2$, and $T2$ are applied to the pseudochallenge $C1$ to produce R_V.
g	The authentic value of the XOR of the prover responses in a given exchange.

B. CHALLENGE EXCHANGE DESCRIPTION

Fig. 4 shows how two parties, each with access to a PUF circuit or its soft model, can perform authentication while only exchanging random ‘pseudo’ challenges. The exchanged challenges are considered ‘pseudo’ challenges because they are never used as direct inputs to the PUF circuit but instead are transformed dynamically before being fed to the PUF circuit. A random pseudochallenge $C1$ is presented by the verifier. The prover receives the pseudochallenge and generates two random pseudochallenges $C2$ and $C3$ such that:

$$PUF\big(Ti(C2)\big) \oplus PUF\big(Tj(C3)\big) = g \qquad (2)$$

Ti and Tj are challenge transformation functions that transform a pseudochallenge into a unique input challenge. The response values of the challenges generated from $C1$ decide which of the transformations are used on the prover pseudochallenges. A number of m exchanges are performed during each authentication session. The value of g is unique for each of the 100 exchanges in the round, as it follows one of the secret g patterns assigned to the device. The pattern selected for g varies from one round to another and is chosen randomly from a set pool of patterns assigned to the device. This choice is hidden from all parties; hence, the verifier must test for all the assigned patterns. To gain access to the PUF device, an authentic party would require both the soft model of the PUF circuit and the unique g patterns it employs.

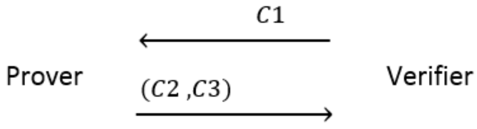

FIGURE 4. The challenge-challenge exchange can authenticate parties by exchanging randomly generated pseudochallenges.

Each device should employ at least two secret patterns. Having a single pattern would significantly reduce the protocol’s security, as this would completely remove the randomness of g. Chosen patterns should also ensure that the assigned value of g for each of the m exchanges in the authentication has an even chance of being ‘0’ or ‘1’. An example of two simple, unique patterns that comply with this is having $g = \{0,0,\dots,0\}$ or $g = \{1,1,\dots,1\}$ for all m exchanges in the round. At the start of each authentication round, the prover randomly selects one of these g patterns to use. Testing performed in Section V shows that two patterns are sufficient for establishing reliable security. However, more patterns can be employed for increased security with little overhead, as the verifier would need to process only a few more bitwise comparisons for each of the added patterns. We also note that since the g patterns are complementary to each other, only half of the patterns need to be stored, while their complementary counterparts can be generated at runtime.

C. COMBINED CHALLENGE GENERATION

Fig. 5 shows how both the verifier and the prover generate a portion of the verification challenge $C1$. For an n-bit challenge, the prover and verifier each generate an (n/2)-bit challenge and exchange them. The verification challenge $C1$ is the concatenation of the generated challenges. By having both parties participate in the generation of the verification challenge, the protocol guarantees each exchange’s freshness. The combined generation process reduces the verification challenge space of the PUF to (n/2) bits, as half of the bits are controlled by the prover. Hence, the protocol uses a 2n-bit input challenge PUF to maintain an effective challenge space of n-bits.

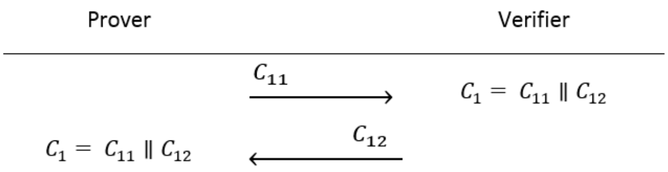
Prover
Verifier
C_{11}
$C_1 = C_{11} \parallel C_{12}$
C_{12}
$C_1 = C_{11} \parallel C_{12}$

FIGURE 5. Verification challenge $C1$ generated by both the prover and verifier using concatenation.

D. CONDITIONAL CHALLENGE TRANSFORMATION

The purpose of the transformation functions is to protect against MITM substitution attacks by forcing the challenges fed to the PUF to be uncorrelated even if an MITM manipulates the exchanged pseudochallenges. An illustration of this attack and the detailed design and analysis of the introduced transformation functions, a significant contribution of this work, are presented in Section IV.

Fig. 6 shows the flowchart for the utilized conditional transformations. The verifier pseudochallenge is transformed into three uncorrelated challenges, each producing a unique response. This results in a 3-bit response R_V, the verifier’s pseudochallenge. R_V is completely hidden from the attacker and used as the decision seed for selecting the transformations applied on $C2$ and $C3$. The conditional transformations hide the actual input challenge of the PUF circuit. We refer to challenges $C1$, $C2$, and $C3$ as ‘pseudo’ challenges, as they are never used as direct inputs for the PUF circuit. In the discussed version of the protocol, we utilize four uncorrelated transformation functions, $T1$, $T2$,

$T3$, and $T4$, to transform the exchanged pseudochallenges into new uncorrelated challenges.

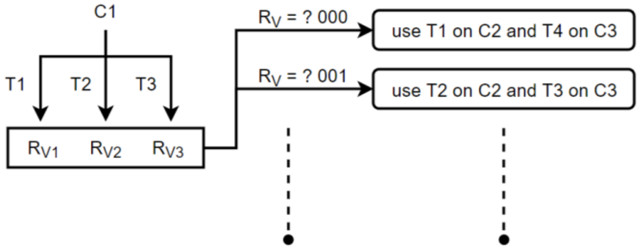

FIGURE 6. The prover challenges C2 and C3 are transformed conditionally based on the verification challenge C1 and the transformed responses.

Table II shows the eight authentication conditions that could be tested by the verifier in a given exchange. The eight unique conditions are a result of applying a unique combination of transformations on $C2$ and $C3$. The transformations, and hence the authentication conditions, are determined by the value of R_V. An adversary impersonating an authentic party would fail to produce the proper pattern of g across the m exchanges. By randomly choosing a pattern from an assigned set of secret patterns, we drastically increase the protocol's security. The pool of g patterns should be unique for each device and should be kept secret.

TABLE II
THE SELECTED PUF EXCHANGE BASED ON THE VALUE OF R_V

R_V			Authentication Condition
R_{V1}	R_{V2}	R_{V3}	$PUF(TiC2) \oplus PUF(TjC3) = g$
0	0	0	$PUF(T1(C2)) \oplus PUF(T4(C3)) = g$
0	0	1	$PUF(T2(C2)) \oplus PUF(T3(C3)) = g$
0	1	0	$PUF(T3(C2)) \oplus PUF(T2(C3)) = g$
0	1	1	$PUF(T4(C2)) \oplus PUF(T1(C3)) = g$
1	0	0	$PUF(T1(C2)) \oplus PUF(T3(C3)) = g$
1	0	1	$PUF(T2(C2)) \oplus PUF(T4(C3)) = g$
1	1	0	$PUF(T3(C2)) \oplus PUF(T1(C3)) = g$
1	1	1	$PUF(T4(C2)) \oplus PUF(T2(C3)) = g$

E. AUTHENTICATION PROTOCOL STEPS

Fig. 7 shows the authentication protocol utilizing the transformed challenge exchange mechanism. The verifier and prover are parties in possession of the PUF circuit or an accurate soft model of the PUF, which can be used to generate responses for any random challenge. As the verification process requires lightweight computation, the constrained device can also play the verifier's role. This mutual authentication feature is one of the main features of the proposed protocol.

In step 1 of the protocol, an initialization message and the PUF identification number are exchanged. The g pattern for this round is selected from a pool of secret patterns assigned to the device. In step 2 of the protocol, the transformed challenge exchange process is performed m times. As the verifier has no access to the value of g, they keep track of $PUF(Ti(C1)) \oplus PUF\big(Tj(C3)\big) = g'$. In step 3 of the protocol, the verifier checks whether the pattern of g' corresponds to one of the patterns assigned to the device. If it does not match any of the assigned patterns, the verifier refuse to authenticate the other party. The protocol can be modified to add some error tolerance by allowing for mismatches in a small portion of the generated g' values. Such a tolerance rate should be calibrated per the utilized PUF's expected error rate.

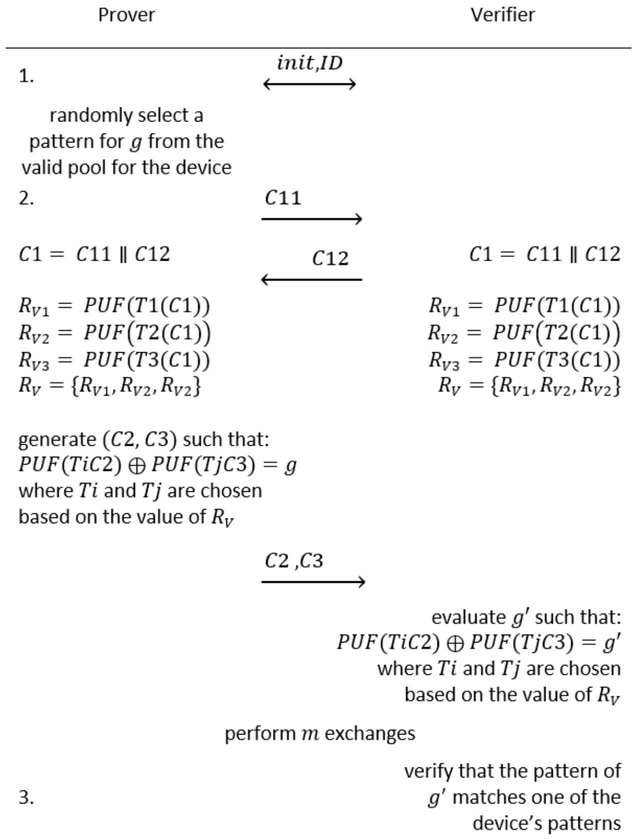
Prover
Verifier
1.
$\xleftrightarrow{init,ID}$
randomly select a pattern for g from the valid pool for the device
2.
$\xrightarrow{C11}$
$C1 = C11 \parallel C12$
$\xleftarrow{C12}$
$C1 = C11 \parallel C12$
$R_{V1} = PUF(T1(C1))$
$R_{V2} = PUF\big(T2(C1)\big)$
$R_{V3} = PUF(T3(C1))$
$R_V = \{R_{V1}, R_{V2}, R_{V2}\}$
$R_{V1} = PUF(T1(C1))$
$R_{V2} = PUF\big(T2(C1)\big)$
$R_{V3} = PUF(T3(C1))$
$R_V = \{R_{V1}, R_{V2}, R_{V2}\}$
generate $(C2, C3)$ such that:
$PUF(TiC2) \oplus PUF(TjC3) = g$
where Ti and Tj are chosen based on the value of R_V
$\xrightarrow{C2\ ,C3}$
evaluate g' such that:
$PUF(TiC2) \oplus PUF(TjC3) = g'$
where Ti and Tj are chosen based on the value of R_V
perform m exchanges
3.
verify that the pattern of g' matches one of the device's patterns

FIGURE 7. Detailed steps of the authentication protocol utilizing the challenge-challenge exchange.

F. SECRET MESSAGE EXCHANGE

One of the features of the LPA protocol is its ability to offer authenticated secret message exchange between trusted parties. By having two possible authentic response values for a fixed value of g, we can encode each authentic response with a different data bit. An adversary could only guess the data bit, as the PUF's response is never exposed. Table III shows how two possible authentic challenge combinations can be encoded with different data bits in an exchange where $g = 0$.

TABLE III
POSSIBLE RESULTS OF SECRET DATA BIT EXCHANGE

$\xrightarrow{C2,C3}$			Authenticity (for $g = 0$)	Data bit d_i
$PUF(Ti(C2))$	$PUF(Tj(C3))$	$\oplus$		
0	0	0	Authentic	0
0	1	1	Rejected	N/A
1	0	1	Rejected	N/A
1	1	0	Authentic	1

By offering secure encryption to highly constrained devices, the introduced protocol dramatically enhances the security of devices and allows for the protocol's use in a wider variety of applications. To the best of our knowledge, the introduced LPA protocol is the first lightweight PUF-based protocol that allows for secure authenticated secret message exchange, where both the authenticity and secrecy of the message are preserved.

IV. DESIGN OF UNCORRELATED CHALLENGE TRANSFORMATION FUNCTIONS

It is desirable to have all the challenge inputs of a PUF circuit be completely uncorrelated, where any two unique challenges have even chances of sharing a response. However, some PUFs, such as the arbiter PUF and its variants, have high correlations among their challenge inputs, and as such, their unique challenges are not necessarily uncorrelated. This was one of the issues addressed in the design of the LS-PUF [34]. This work introduces a novel method for producing uncorrelated challenges when using the arbiter PUF and its variants, such as the XOR-PUF, LS-PUF, FF-PUF, or differential amplifier PUF (DA-PUF) [38]. We note that using these PUFs is not required by the protocol; we introduce this method to illustrate how these popular PUFs with naturally high correlations between their input challenges can still be used in our protocol with very little overhead. The uncorrelated transformations can force the PUF circuit's challenges to be decorrelated even if a third party manipulates the challenges.

A. INPUT CHALLENGE RESPONSE CORRELATION

In [39], the impact of introducing a single bit flip in the PUF input challenge of an arbiter PUF was investigated. We verify and reproduce this work using the arbiter PUF's mathematical model. We use the same delay model utilized in [39], which is based on the delay variations presented in [40]. The results are shown in Fig. 8. The input bit flips introduced near the middle of the PUF circuit have a 50% chance of producing a bit flip at the PUF output.

Prior to this work, the effect of multiple induced challenge bit flips on arbiter PUFs was not adequately examined. We observe that introducing multiple bit flips has a drastically different effect than that of a single bit flip. For instance, when using the same mathematical model to produce Fig. 8, introducing two bit flips at locations 31 and 33 would result in a 10% probability of obtaining a response bit flip. This very low probability is contrary to the intuition of Fig. 8, where both bit flips, show a probability very close to 50% when introduced alone.

To better understand the PUF output correlations when multiple bit flips are introduced, we examine a case study where five instances of input bit flips are introduced into a 64-bit PUF circuit. Fig. 9 shows the effects and locations of the introduced bit flip instances labeled A, B, C, D, and E on a 64-bit arbiter PUF circuit. The shaded areas in each instance show the portion of the PUF where the delay elements have swapped paths in the circuit (upper vs. lower). The delay elements are either connected to the same path as that before the induced bit flips (nonshaded) or swap paths due to the induced bit flips (shaded). We conjecture that the probability *Ps* of two instances of induced bit flips sharing the same PUF response value can be predicted by the ratio of matching delay element connections to the total number of delay elements. To verify our conjectured estimation method, we use the mathematical model of the arbiter PUF and evaluate the output random challenges with the bit flip instances shown in Fig. 9. A total of 200 PUF instances are produced, and 5000 challenges are tested in each instance. Table IV shows the experimental results of the bit flips along with the ratio of the shared delay elements.

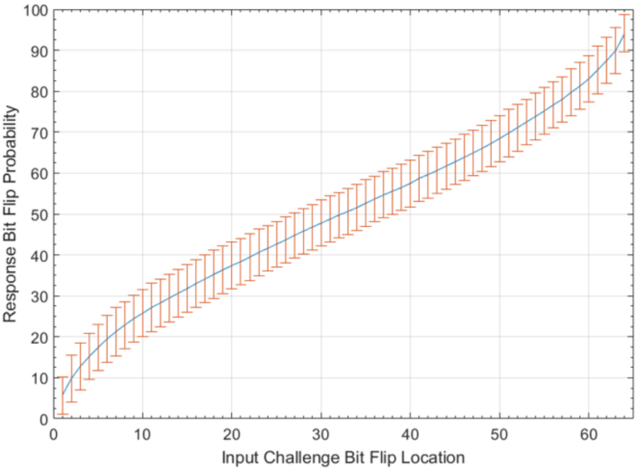

FIGURE 8. **Probability of a response bit flip due to the single input challenge bit flip of a 64-bit arbiter PUF.**

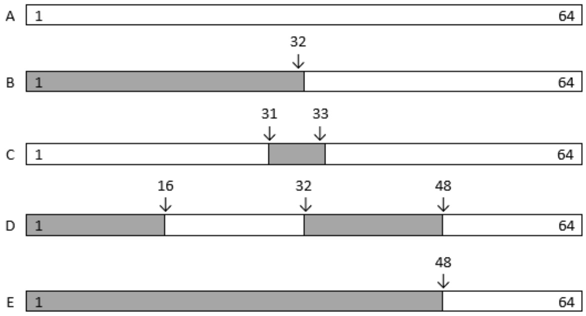

FIGURE 9. **The tested bit flip instances and their effect on the delay element path connections. Shaded areas represent delay elements that have swapped paths.**

TABLE IV
THE TESTED INSTANCES REGARDING THE MATCHING CONNECTION RATIO AND THE PROBABILITY OF PRODUCING THE SAME OUTPUT RESPONSE BIT

	Correlations with instance A			
	AB	AC	AD	AE
Mismatch Ratio	0.50	0.95	0.5	0.25
Probability P_S	0.505	0.885	0.501	0.338

The results show that the probability of introducing a response bit flip is quasi-proportional to the ratio of shared delay elements (shared shading).

The complete relationship between the matching connection ratio and output response is shown in Fig. 10. The plot is produced by introducing multiple bit flips at random locations and examining the response correlation with various challenges. We can see that an uncorrelated response can always be produced by introducing challenge bit flips that result in a 0.5 ratio of mismatched delay elements. This insight is used to design the four uncorrelated transformation functions utilized by LPA.

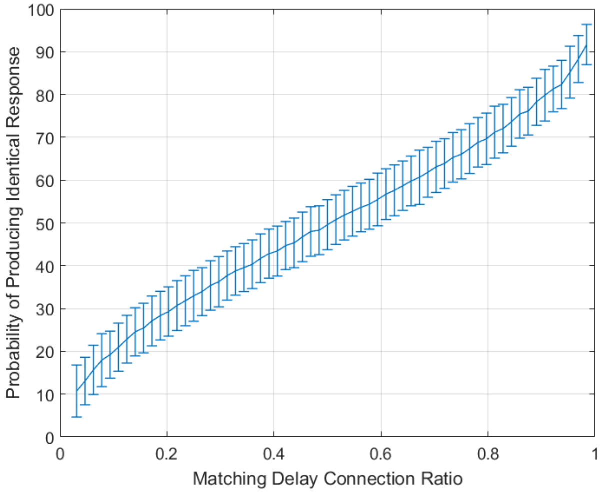

FIGURE 10. Probability of response bit flip correlation with the ratio of matching delay elements when inducing multiple challenge bit flips.

B. TRANSFORMATION FUNCTIONS

Fig. 11 illustrates how the padding bits chosen for the transformation functions could result in four unique circuit connections, where each has a matching delay element connection (shaded/unshaded) ratio of 0.5 relative to any other transformation instance. This ensures a lack of correlations between the transformed challenges. For a protocol utilizing 128-bit pseudochallenges and a 131-bit arbiter PUF circuit, these locations would be 34, 65, and 97. The correlations among the transformation functions are verified using the mathematical model of the arbiter PUF. The verification result is shown in Table V, where 200 arbiter PUF instances are fed 5000 random pseudochallenges. The functions utilized are uncorrelated, as the probability of producing the same response is very close to 0.5, as the results show in the table.

TABLE V
TRANSFORMATION FUNCTION CROSS-CORRELATION PROBABILITIES

Transformation	T2	T3	T4

T1	μ=0.499
	σ=0.042
		μ=0.487
		σ=0.046
			μ=0.498
			σ=0.039

T2		μ=0.497
		σ=0.038
			μ=0.486
			σ=0.044

T3			μ=0.498
			σ=0.041

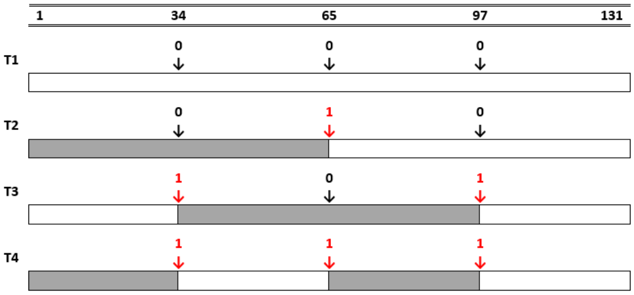

FIGURE 11. The transformation functions T1-T4 are designed by adding additional stages in the arbiter PUF and inserting bit flips at sensitive locations that ensure output response decorrelation.

C. SECURITY IMPACT

The uncorrelated transformation functions introduced are used to add essential protection against advanced MITM attacks. As constrained devices cannot utilize hash functions to guarantee message integrity, PUF exchanges are often vulnerable to substitution attacks. An attacker can substitute one of the communicated challenges with an altered challenge that can expose information about the PUF circuit.

For instance, consider the exchange shown in Fig. 12, where the prover sends the verifier two authentic challenges $C2$ and $C3$. If the verifier utilizes an XOR function on the responses of the untransformed versions of $C2$ and $C3$, then the result of the XOR would always be '0'. While the verifier can impose a check to ensure that $C2 \neq C3$, this check can be bypassed by an attacker who can utilize a challenge $C3$ that shares a high correlation with $C2$ rather than utilizing a challenge that is identical to $C2$. Such an attack would allow the attacker to experiment with challenges and extract information about the authentication session.

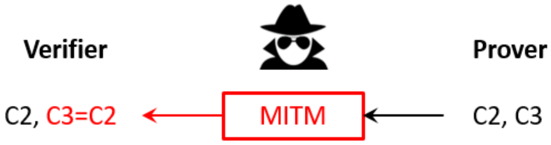

FIGURE 12. An MITM attack can substitute exchanged challenges with malicious ones to extract more information from the circuit by forcing special conditions. In this example, C3 is replaced with a duplicate value of C2.

The purpose of the transformations is to transform a raw challenge Ci into two (or more) challenges $T1(Ci)$ and $T2(Ci)$ such that the probability of $T1(Ci)$ and $T2(Ci)$ sharing the same PUF response is very close to 0.5. By utilizing two distinct uncorrelated functions on the prover challenges, we secure the exchange against the aforementioned MITM attacks by forcing the prover challenges to be uncorrelated regardless of manipulations or substitutions performed by an MITM adversary. The introduced transformation functions are an important contribution of this work, as any protocol can employ them to provide protection against similar MITM attacks that aim to exploit the highly correlated input challenges of arbiter-based PUFs.

V. PROTOCOL SECURITY ANALYSIS

This section investigates the protocol’s security against random guess attacks, replay attacks, DoS attacks, and modeling attacks. The modeling attacks include covariance matrix adaptation with ES (CMA-ES), an ANN, and an SVM. The protocol employs the following:

- An arbiter PUF circuit with $n = 131$ for the input challenge bit width.
- A number of $m = 100$ exchanges per authentication round.
- Two secret g patterns that are complementary to each other are assigned to the device.

For simplicity, we ignore the error margin α, as it can be addressed by increasing the number of exchanges m. This is shown in Section VI, which examines the effect of the PUF error rate on protocol security.

A. RANDOM GUESS ATTACK

In a random guess attack, the attacker simply responds with randomly generated pseudochallenges. The probability of success for a random guess attack is 2^{-m}. With m=100, this protocol provides sufficient security for most applications, as the attack does not expose any information about the PUF.

B. REPLAY ATTACKS

In a replay attack, an attacker collects observed authentic rounds in the hope of utilizing them in a future exchange if presented with some of the collected verification challenges again. However, this attack is infeasible as the number of unique rounds is very large. In the LPA protocol, the verifier party presents the prover party with 100 random challenges. With the verifier controlling 64 bits in each challenge, there are 2^{6400} unique rounds that can be produced by the verifier. The probability of any collected round being presented again by the verifier party is 2^{-6400}. Storing any significant portion of the verification space is entirely infeasible.

C. DENIAL OF SERVICE ATTACKS

In a DoS attack, the adversary aims to temporarily or permanently disrupt the communication of the PUF device. We are concerned with DoS attacks that result in permanent disruptions of device communications. Such attacks can be dangerous, as they can render the device useless even after the adversary ceases their disruption. Protocols that impose a hard limit on the number of authentications that can be performed by the device, such as Lockdown [28], are vulnerable to DoS attacks in which an adversary sends fake authentication requests to exhaust the number of authentications supported by the device.

Protocols that require strict synchronization between parties could also fall victim to DoS attacks that can permanently disable a device. For example, in the protocol presented in [30], the PUF device needs to be in perfect synchronization with the server. A loss of synchronization might result in the complete loss of the device. The introduced LPA protocol is impervious to such DoS attacks, as it imposes no limit on the number of authentications and does not require any synchronization between the parties.

B. MODELING ATTACKS

Resilience against machine learning-based attacks has been the primary benchmark used for evaluating the security of pseudocryptographic PUF-based authentication protocols. The simple arbiter PUF is vulnerable to attacks such as linear regression [10], [11]. In [13], an attack showing increased effectiveness against obfuscated PUF circuits was introduced. More powerful attacks that utilize ANNs or CMA-ES were successfully used in [41], [42] to compromise various PUF-based protocols. This section evaluates the security of the introduced protocol and showcases its high resilience against CMA-ES-, ANN-, and SVM-based attacks. The attacker is assumed to have access to the communication channel between the verifier and the prover, allowing them to observe and collect authentic exchanges.

Tests have shown that the LPA’s suggested configuration offers exceptionally high resilience against machine learning attacks. The protocol can also scale up its security with simple adjustments. For instance, a more secure circuit, such as a 3-XOR FF-PUF, could be used instead of the arbiter PUF, which is considered the most vulnerable among the strong PUF circuits. The number of patterns utilized by each device can also be increased beyond two, which is the minimum required amount. While advancements in computing technology and machine learning techniques might expose the LPA protocol to new threats, the LPA’s high resilience against machine learning attacks would put it in an advantageous position compared to other lightweight PUF-based protocols that will face similar future threats. To the best of our knowledge, the introduced protocol is the first lightweight PUF-based mutual authentication protocol to show high resilience against machine learning attacks while providing authenticated secret message exchange.

1) EVOLUTION STRATEGIES

ES attacks can be performed without direct access to the challenge-response pairs by simply treating the system as a black box. ES attacks generate models and test their ‘fitness’ according to the collected authentication rounds. To evaluate the security of the proposed protocol against ES attacks, we perform CMA-ES using a local implementation of the algorithm presented in [43]. We test the protocol with varying arbiter PUF sizes, ranging from 16-bit PUFs to 131-bit PUFs, while also varying the number of secret g-masks utilized. We also test the security of the protocol with the g-masks exposed.

Fig. 13 shows the accuracy achieved after running the CMA-ES attack for 120 generations on different protocol versions. The size of the arbiter PUF varies between 16 bits and 128 bits. The number of g patterns varies between two secret patterns and eight secret patterns. The secret pattern values are incorporated into the black box search. We also test the effectiveness of CMA-ES attacks when two exposure patterns are utilized. We note that the g patterns should never be exposed. The test against exposed patterns is performed for the sake of analyzing their security impact. A dataset containing three million challenges, corresponding to 10,000 authentication rounds, is used in all the tests.

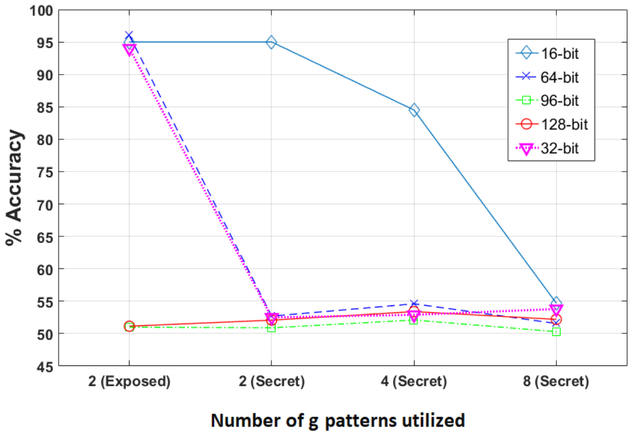

FIGURE 13. Accuracy levels achieved by the CMA-ES attacks against LPA.

Exposing the g-masks makes some of the protocol versions vulnerable to the CMA-ES attack. These vulnerable versions utilize arbiter PUFs that are smaller than 96 bits. The CMA-ES attack is not successful when using 96-bit or 128-bit arbiter PUFs, even when the g patterns are exposed. Hiding the secret patterns drastically increases the security of the protocol, with all versions becoming resilient against the attack except that using a 16-bit arbiter PUF. The tests also show that increasing the number of secret patterns increases the protocol's security, as shown in the figure for the 16-bit arbiter PUF.

It is recommended to use at least a 128-bit arbiter PUF to avoid exhausting the verifier challenge space, as the verifier party controls only half of the input challenge bits for the verification challenge. Using small arbiter PUF circuits (16-bit and 32-bit) might also open up the possibility of brute-forcing the PUF model due to the low uniqueness levels of such small circuits.

In the implemented version of the protocol, we use a 131-bit arbiter PUF and two secret patterns. As the transformation functions use three bits of the input challenge, the resultant effective challenge space contains 128 bits. CMA-ES attacks are performed with three million collected challenges. The patterns are exposed in these attacks and hence are not part of the CMA-ES search. This is done to increase the confidence in the protocol's resilience. A CMA-ES attack is also performed on Slender [27] to highlight the introduced protocol's increased security. Table VI shows the number of collected challenges and the corresponding number of authentication sessions observed for the ES attacks on Slender and LPA.

TABLE VI
NUMBER OF CHALLENGES AND OBSERVED AUTHENTICATION SESSIONS UTILIZED IN THE CMA-ES ATTACKS ON SLENDER AND LPA.

	Challenges Collected	Authentication Sessions Observed
Slender	1.25 *million*	1000
LPA	3 *million*	10,000

Fig. 14 shows the achieved model accuracy progression across several runs over 120 generations of ES attacks on Slender (blue-dashed) and the proposed LPA protocol (red-solid). All the attacks on the LPA protocol fail to achieving any noticeable accuracy gains despite the runs being initialized with relatively high accuracy. These results show that the introduced protocol is highly resilient against CMA-ES attacks even when the secret *g* patterns associated with the device are exposed.

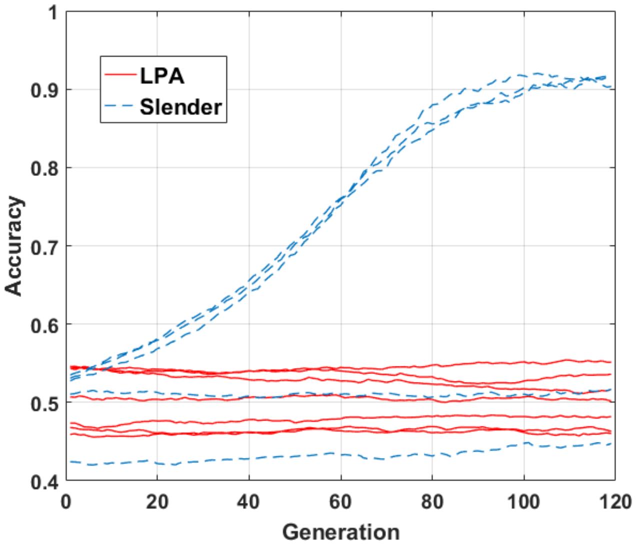

FIGURE 14. Accuracy levels achieved by the CMA-ES attacks on Slender (blue-dashed) and the introduced LPA protocol (red-solid).

2) ANN AND SVM ATTACKS

ANNs) are used to compromise the security of the PUF circuits and protocols developed in [44], [45]. We test the performance of the ANN on the LPA protocol to show its resilience to this modeling method. A single exchange-based classification is impossible to execute due to the randomness of individual exchanges. Hence, the ANN must be trained to distinguish between complete authentic rounds and nonauthentic rounds rather than distinguishing between individual exchanges.

Testing has shown that it is infeasible to perform this classification when collecting up to 50,000 authentication rounds for training. However, we note that even if successful, such a trained network would not pose a risk to the protocol, as it would simply test whether a set of 300 challenges might pass authentication. This trained neural network is of no practical use to an attacker for several reasons:

1) The trained ANN would only allow the attacker to

distinguish between rounds that have already been completed, i.e., all the verifier challenges that have been presented. However, in a live round, the prover must respond to each verifier challenge before receiving the next challenge.

2) Attempting to store possible authentic rounds offline would also be infeasible. With 100 challenges per round and 64 bits per challenge (controlled by the verifier), there would be 2^{6400} unique verification rounds. Storing any significant portion of the rounds would be impractical.

3) Even when presented with all the verifier challenges, attempting to generate a single authentic round would be computationally infeasible. The attacker would need to repeatedly generate 100 random challenges until one set of 100 challenges passes the trained ANN classification test. Such an attempt would require an average of 2^{100} repetitions.

For these reasons, we consider this ANN attack to pose no risk to the protocol. However, we test the effectiveness of such an ANN for the sake of the completeness of this work. The results show that an ANN cannot distinguish between authentic and nonauthentic rounds when trained with tens of thousands of collected authentications. Fifty thousand authentic rounds are combined with 50,000 nonauthentic rounds. The dataset is divided into a training set (9 million exchanges or 90,000 authentication rounds) and a test set (1 million exchange or 10,000 authentication rounds). The dataset is repeatedly shuffled and split to ensure that the test set has a minimal bias towards either label value (authentic vs. nonauthentic).

We use multiple ANN architectures and restart the classification 50 times to observe the maximum, minimum, and average accuracy rates. The number of hidden layers is varied from 2 to 4. The average number of nodes per layer is varied from 20 to 120 in increments of 20. The results show that the ANN attack fails to achieve any accuracy gains, as shown in Table VII. The attacks are implemented in TensorFlow utilizing the Keras framework. The rectified linear unit (ReLU) function is used for the hidden layers, and the sigmoid function is used for the output layer. We use the Adam optimizer [46] to update the weights and the binary cross-entropy function as the loss function. L2 regularization is used for the runs shown in Table VII. The mean accuracy remains at 0.5 across all restarts for all architectures. Test runs with no regularization are also performed, and they yield a mean test accuracy of 0.500.

TABLE VII
SUMMARY OF ANN ATTACKS ON LPA

Number of Hidden Layers	Average Number of Nodes Per Layer	ANN Training Results	
		Mean Test Accuracy	Max Test Accuracy
2	20 to 120	0.5000	0.5002
3		0.5000	0.5002
4		0.5000	0.5002

Classification using an SVM shows similar results, although the training process is limited to a dataset of 10,000 authentication rounds. Training with a larger dataset proves problematic, as it exceeds our system's memory capacity of 32 GB of RAM. Multiple SVM kernels are tested, and all show no measurable accuracy gains. The GPU-accelerated library ThunderSVM [47] is used to perform SVM training. Restarts are performed by shuffling the dataset before slicing it again into training and testing sets. Table VIII summarizes the SVM attack results, showing the mean accuracy and standard deviation across all restarts.

TABLE VIII
RESULT OF SVM ATTACKS USING 10000 AUTHENTICATION SESSIONS

Kernel	Mean Accuracy	STD
RBF	0.500	0.004
Polynomial	0.500	0.001
Sigmoid	0.500	0.004
Linear	0.500	0.004

With no measurable accuracy gains observed, the feasibility of the suggested classifier is questionable. More importantly, this classifier poses a minimal risk, as it cannot compromise the protocol even when successfully trained.

VI. PUF CIRCUIT ERROR RATE AND TUNING

Due to the erroneous outputs of PUF circuits, the authentication protocol requires the employment of some error tolerance value α. This value corresponds to the number of exchanges that are allowed to fall out of the selected *g* pattern within a given round of *m* exchanges. Tolerance would allow an attacker to have a higher chance of guessing a response. To counter this increased guess chance and maintain the original security level, the total number of exchanges *m* in a round can be increased. A simple search script is used to find the minimum value of *m* that maintains the desired random guess probability while supporting the required error tolerance rate. The results are shown in Fig. 15.

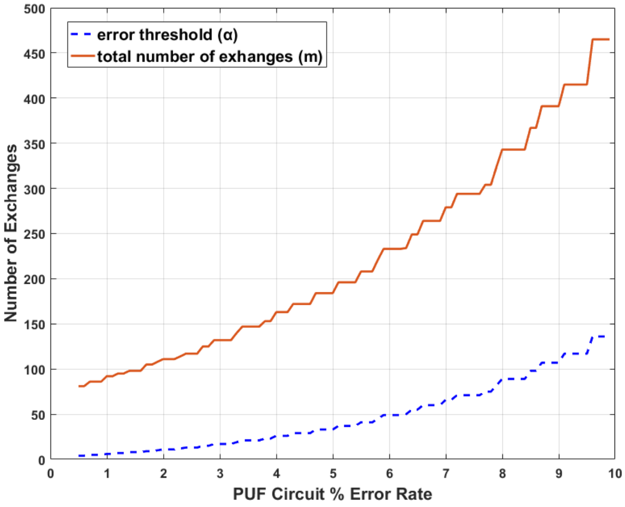

FIGURE 15. The required growth of m and α to compensate for the PUF circuit error rate while maintaining a 64-bit level of security.

TABLE IX
NUMBER OF EXCHANGES REQUIRED TO MAINTAIN 64-BIT SECURITY

% PUF Circuit Error	% Faulty Exchanges	% Fault Tolerance	# Exchanges m	Fault Tolerance Threshold α
$E_C = 0.5\%$	$E_A = 1.7\%$	$t = 4.7\%$	$m = 81$	$\alpha = 4$
$E_C = 4\%$	$E_A = 12.6\%$	$t = 15.6\%$	$m = 163$	$\alpha = 26$
$E_C = 8\%$	$E_A = 22.5\%$	$t = 25.5\%$	$m = 343$	$\alpha = 89$

The values of m and α can be increased to compensate for the PUF circuit error while maintaining a 64-bit security level. The fault tolerance value α is chosen to be 3% higher than the value of the expected error rate of authentic exchanges E_A. Table IX highlights the tuning values of α and m at some key values of the PUF error rate. The table shows the value of the PUF circuit error rate E_C, the expected error rate for the authentic exchanges E_A, the tolerance rate t, the number of exchanges m that need to be utilized to maintain 64-bit security, and the number of allowed erroneous exchanges α.

From the table, we can see that the m and α values remain manageable even when utilizing circuits with up to an 8% error rate. The values can be tuned dynamically depending on the operating conditions and the aging effect of the circuit. We note that higher error rates can be problematic when using the LPA protocol for secret message exchange. It is advisable to use a highly reliable PUF circuit in addition to error correction or detection when deploying the protocol for secret message exchange.

VII. DELAY AND THROUGHPUT ANALYSIS

We present a throughput and delay analysis for the proposed protocol. The throughput and delay values can vary depending on the fabrication technology used, the PUF circuit chosen, and the random number generator (RNG) chosen. In our implementation, a simple linear feedback shift register (LFSR) circuit is used to generate random challenges. The protocol logic circuit is implemented in 45 nm ASIC technology using Cadence. The implemented circuit can run at frequencies up to 1300 MHz. Verifying an exchange requires 17 clock cycles, while producing a prover response requires 21 clock cycles. This results in a delay of 15 ns for verifying an exchange and 18 ns for generating the prover response.

These delay numbers assume that the PUF circuit can generate a response bit in one clock cycle. However, for many PUF circuits, this is not the case. The implemented circuit is designed to wait on the PUF output when needed. Depending on the speed of the PUF circuit, this could increase the total delay incurred by the exchange. A 131-bit arbiter PUF implemented in 45 nm technology shows a delay of 9.77 ns per evaluation. The implemented PUF circuit is designed to signal the control module when the PUF response is ready for reading. When utilized with our protocol, this yields an average processing time of 73 ns on the prover side. Overall, the processing delay at the prover side is 7.3 µs for an authentication round that utilizes 100 exchanges.

These delay figures illustrate the feasibility of a circuit implementation that can process an exchange promptly. However, as the final delay and throughput values depend on various factors, such as the chosen communication technology, PUF circuit, and fabrication technology, we present a more generalized delay and throughput characterization of the protocol in Table X. The table shows the average number of PUF evaluations required, the average number of challenges generated, and the amount of data transmitted by each side during an authentication round with 100 exchanges.

TABLE X
THROUGHPUT STATS FOR AN LPA PROTOCOL AUTHENTICATION ROUND

	Verifier Side	Prover Side
No. of PUF evaluations	500	600
No. of challenges generated	100	300
Data transferred (kilobytes)	1.6 kB	3.2 kB

VIII. RELATED WORK

While PUFs are gaining more popularity in security applications, only a handful of lightweight protocols that avoid cryptographic or hash functions, have been introduced so far. Protocols that utilize hashing or encryption [13]–[25] provide solid resilience against modeling attacks. However, this comes at the cost of an increased hardware implementation area that might not be suitable for small devices.

Several lightweight protocols that employ CRP obfuscation techniques have been suggested [27]–[32]. However, none offer authenticated secret message exchange, while some are vulnerable to modeling attacks, as with the method of [27]. Table XI lists the requirements and features of the LPA protocol, along with those of the other mentioned lightweight protocols. The introduced LPA protocol has the unique feature of offering authenticated secret message exchange. The requirements and features listed in the table include the following:

- The requirement of a true RNG (TRNG).
- The number of authentications supported: hard limit (*l*), delay-based/throughput limit (*d*), or infinity.
- Support for mutual authentication.
- Support for secret message exchange. This requirement is different from secret key sharing, which only allows arbitrary values to be communicated.

TABLE XI
LIGHTWEIGHT PUF-BASED SECURITY PROTOCOL COMPARISON

Protocol	TRNG Req.	# Auth.	Mutual Auth.	Secret Mess.
Slender [27]	✓	∞	×	×
Lockdown [28]	✓	*l*	✓	×
Obfuscated [29]	✓	∞	×	×
Zalivaka [30]	×	∞	×	×

Gu [31]	✓	*d*	✓	×
Zhang [32]	✓	∞	×	×
LPA (this work)	✓	∞	✓	✓

Table XII shows a comparison between the throughput and communication requirements of LPA and the estimates of these requirements for other lightweight PUF-based protocols. From the table, we can see that the introduced LPA protocol requires the communication of more bits on the prover side than other protocols. This tradeoff is acceptable for devices that require high resilience or demand secret message exchange, as the LPA protocol is the only lightweight PUF protocol that offers these features. The number of bits transmitted in the LPA protocol can be drastically reduced by transmitting a seed for generating the challenges instead of transmitting the challenges themselves. We leave such enhancement to future work.

TABLE XII
COMPARISON OF THROUGHPUT REQUIREMENTS AT THE PROVER SIDE

Protocol	# PUF Evaluations	Bits Transmitted
Slender [27]	1250	2500
Lockdown [28]	1000	1000
Gu [31]	64	64
LPA (this work)	600	26200

IX. CONCLUSION

We introduce an LPA protocol based on secret pattern recognition. A party's authenticity is verified by checking for a set of unique, secret exchange patterns assigned to the device. The protocol's resilience against machine learning attacks is demonstrated by performing simulated modeling attacks using CMA-ES, ANNs, and SVMs. The results show that the proposed protocol exhibits very high resilience against modeling attacks even when the secret patterns associated with the device are exposed. The protocol offers security features such as mutual authentication and authenticated secret message exchange, which are currently not offered by any lightweight PUF-based protocol. A method for resisting MITM attacks that aim to exploit challenge correlations is also introduced. The LPA protocol provides constrained IoT devices with additional security features that allow for their deployment in a wider variety of applications. A throughput enhancement for the protocol is planned for future work, in which we will explore techniques for increasing its communication efficiency.

REFERENCES

[1] H. Chien, "SASI: A new ultralightweight rfid authentication protocol providing strong authentication and strong integrity," *IEEE Trans. Dependable Sec. Comput.*, vol. 4, no. 4, pp. 337–340, Oct. 2007, doi: 10.1109/TDSC.2007.70226.

[2] S. Sarma, "Towards the five-cent tag." Technical Report MIT-AUTOID-WH-006, MIT Auto ID Center. 2001. Available: http://www.autoidlabs.org

[3] V. Boyko, P. MacKenzie, and S. Patel, "Provably secure password-authenticated key exchange using Diffie-Hellman," in *Proc. 19th Int. Conf. Theory Appl. Cryptographic Techn.*, Bruges, Belgium: Springer-Verlag, 2000, pp. 156–171.

[4] M. Bellare, D. Pointcheval, and P. Rogaway, "Authenticated key exchange secure against dictionary attacks," in *Int. Conf. Theory Appl. Cryptographic Techn.*, Berlin, Heidelberg: Springer, 2000, pp. 139–155.

[5] D. Liu, Z. Liu, Z. Yong, X. Zou, and J. Cheng, "Design and implementation of an ECC-based digital baseband controller for RFID tag chip," *IEEE Trans. Ind. Electron.*, vol. 62, no. 7, pp. 4365–4373, Jul. 2015, doi: 10.1109/TIE.2014.2387333.

[6] B. Gassend, D. Clarke, M. V. Dijk, and S. Devadas, "Silicon physical random functions," in *Proc. 9th ACM Conf. Comput. Commun. Secur.*, Washington, DC, USA: Association for Computing Machinery, 2002, pp. 148–160.

[7] N. A. Anagnostopoulos, S. Ahmad, T. Arul, D. Steinmetzer, M. Hollick, and S. Katzenbeisser, "Low-cost security for next-generation IoT networks," *ACM Trans. Internet Technol.*, vol. 20, no. 3, p. 30, Sept. 2020, doi: 10.1145/3406280.

[8] D. Rizk, R. Rizk, and S. Hsu, "Applied layered-security model to IoMT," in *2019 IEEE Int. Conf. Intell. Secur. Inf. (ISI)*, Shenzhen, China: IEEE, 2019, pp. 227–227.

[9] J. Guajardo, S. S. Kumar, G. J. Schrijen, and P. Tuyls, "FPGA intrinsic PUFs and their use for IP protection," in *Int. Workshop Cryptographic Hardware Embedded Syst.*, Berlin, Heidelberg: Springer, 2007, pp. 63–80.

[10] U. Rührmair *et al.*, "PUF modeling attacks on simulated and silicon data," *IEEE Trans. Inf. Forensics Security*, vol. 8, no. 11, pp. 1876–1891, Nov. 2013, doi: 10.1109/TIFS.2013.2279798.

[11] J. Tobisch and G. T. Becker, "On the scaling of machine learning attacks on PUFs with application to noise bifurcation," in *Int. Workshop Radio Freq. Identification: Secur. Privacy Issues*, Cham: Springer International Publishing, 2015, pp. 17–31.

[12] F. Ganji, S. Tajik, and J. P. Seifert, "Why attackers win: On the learnability of XOR arbiter PUFs," in *Int. Conf. Trust Trustworthy Comput.*, Cham: Springer International Publishing, 2015, pp. 22–39.

[13] J. Shi, Y. Lu, and J. Zhang, "Approximation attacks on strong PUFs," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 39, no. 10, pp. 2138–2151, Oct. 2020, doi: 10.1109/TCAD.2019.2962115.

[14] B. Gassend, D. Clarke, M. V. Dijk, and S. Devadas, "Security with noisy data," in *Controlled Phys. Random Functions*, London: Springer, 2002, pp. 235–253.

[15] A. R. Sadeghi, I. Visconti, and C. Wachsmann, "PUF-enhanced RFID security and privacy," in *Workshop Secure Compon. Syst. Identification (SECSI)*: Cologne, Germany, 2010,

[16] A. Van Herrewege *et al.*, "Reverse fuzzy extractors: Enabling lightweight mutual authentication for PUF-enabled RFIDs," in *Int.*

Conf. Financial Cryptography Data Secur., Berlin, Heidelberg: Springer, 2012, pp. 374–389.

[17] P. Gope, A. K. Das, N. Kumar, and Y. Cheng, "Lightweight and physically secure anonymous mutual authentication protocol for real-time data access in industrial wireless sensor networks," *IEEE Trans. Ind. Informat.*, vol. 15, no. 9, pp. 4957–4968, Sept. 2019, doi: 10.1109/TII.2019.2895030.

[18] U. Chatterjee *et al.*, "Building PUF based authentication and key exchange protocol for iot without explicit CRPs in verifier database," *IEEE Trans. Dependable Sec. Comput.*, vol. 16, no. 3, pp. 424–437, May 2019, doi: 10.1109/TDSC.2018.2832201.

[19] V. P. Yanambaka, S. P. Mohanty, E. Kougianos, and D. Puthal, "PMsec: Physical unclonable function-based robust and lightweight authentication in the internet of medical things," *IEEE Trans. Consum. Electron.*, vol. 65, no. 3, pp. 388–397, Aug. 2019, doi: 10.1109/TCE.2019.2926192.

[20] Y. Yilmaz, V. Do, and B. Halak, "ARMOR: An anti-counteRfeit security mechanism for lOw cost radio frequency identification systems," *IEEE Trans. Emerg. Topics Comput.*, pp. 1–1, Jan. 2020, doi: 10.1109/TETC.2020.2964435.

[21] M. A. Qureshi and A. Munir, "PUF-IPA: A PUF-based identity preserving protocol for internet of things authentication," in *2020 IEEE 17th Annu. Consum. Commun. Netw. Conf. (CCNC)*, Las Vegas, NV, USA: IEEE, 2020, pp. 1–7.

[22] J. Long, W. Liang, K. C. Li, D. Zhang, M. Tang, and H. Luo, "PUF-based anonymous authentication scheme for hardware devices and IPs in edge computing environment," *IEEE Access*, vol. 7, pp. 124785–124796, Jun. 2019, doi: 10.1109/ACCESS.2019.2925106.

[23] Y. Chen, W. Kong, and X. Jiang, "Anti-synchronization and robust authentication for noisy PUF-based smart card," *IEEE Access*, vol. 7, pp. 142214–142223, Sept. 2019, doi: 10.1109/ACCESS.2019.2944515.

[24] S. Li, T. Zhang, B. Yu, and K. He, "A provably secure and practical PUF-based end-to-end mutual authentication and key exchange protocol for IoT," *IEEE Sensors J.*, vol. 21, no. 4, pp. 5487–5501, Feb. 2021, doi: 10.1109/JSEN.2020.3028872.

[25] J. W. Byun, "PDAKE: A provably secure PUF-based device authenticated key exchange in cloud setting," *IEEE Access*, vol. 7, pp. 181165–181177, Dec. 2019, doi: 10.1109/ACCESS.2019.2957742.

[26] Y. Gao, Y. Su, L. Xu, and D. C. Ranasinghe, "Lightweight (Reverse) fuzzy extractor with multiple reference PUF responses," *IEEE Trans. Inf. Forensics Security*, vol. 14, no. 7, pp. 1887–1901, Dec. 2019, doi: 10.1109/TIFS.2018.2886624.

[27] M. Majzoobi, M. Rostami, F. Koushanfar, D. S. Wallach, and S. Devadas, "Slender PUF protocol: A lightweight, robust, and secure authentication by substring matching," in *2012 IEEE Symp. Secur. Privacy Workshops*, San Francisco, CA, USA: IEEE, 2012, pp. 33–44.

[28] M. Yu, M. Hiller, J. Delvaux, R. Sowell, S. Devadas, and I. Verbauwhede, "A lockdown technique to prevent machine learning on PUFs for lightweight authentication," *IEEE Trans. Multi-Scale Comput. Syst.*, vol. 2, no. 3, pp. 146–159, Apr. 2016, doi: 10.1109/TMSCS.2016.2553027.

[29] Y. Gao *et al.*, "Obfuscated challenge-response: A secure lightweight authentication mechanism for PUF-based pervasive devices," in *2016 IEEE Int. Conf. Pervasive Comput. Commun. Workshops (PerCom Workshops)*, Sydney, NSW, Australia: IEEE, 2016, pp. 1–6.

[30] S. S. Zalivaka, A. A. Ivaniuk, and C. Chang, "Reliable and modeling attack resistant authentication of arbiter PUF in FPGA implementation with trinary quadruple response," *IEEE Trans. Inf. Forensics Security*, vol. 14, no. 4, pp. 1109–1123, Sept. 2019, doi: 10.1109/TIFS.2018.2870835.

[31] 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 Trans. Comput.-Aided Design Integr. Circuits Syst.*, pp. 1–1, Nov. 2020, doi: 10.1109/TCAD.2020.3036807.

[32] J. Zhang and C. Shen, "Set-based obfuscation for strong PUFs against machine learning attacks," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 68, no. 1, pp. 288–300, Oct. 2021, doi: 10.1109/TCSI.2020.3028508.

[33] G. E. Suh and S. Devadas, "Physical unclonable functions for device authentication and secret key generation," in *2007 44th ACM/IEEE Des. Automat. Conf.*, San Diego, CA, USA: IEEE, 2007, pp. 9–14.

[34] M. Majzoobi, F. Koushanfar, and M. Potkonjak, "Lightweight secure PUFs," in *2008 IEEE/ACM Int. Conf. Comput.-Aided Des.*, San Jose, CA, USA: IEEE, 2008, pp. 670–673.

[35] J. W. Lee, L. Daihyun, B. Gassend, G. E. Suh, M. V. Dijk, and S. Devadas, "A technique to build a secret key in integrated circuits for identification and authentication applications," in *2004 Symp. VLSI Circuits. Dig. Tech. Papers (IEEE Cat. No.04CH37525)*, Honolulu, HI, USA: IEEE, 2004, pp. 176–179.

[36] S. V. S. Avvaru, Z. Zeng, and K. K. Parhi, "Homogeneous and heterogeneous feed-forward XOR physical unclonable functions," *IEEE Trans. Inf. Forensics Security*, vol. 15, pp. 2485–2498, Jan. 2020, doi: 10.1109/TIFS.2020.2968113.

[37] T. Idriss, H. Idriss, and M. Bayoumi, "A PUF-based paradigm for IoT security," in *2016 IEEE 3rd World Forum Internet Things (WF-IoT)*, Reston, VA, USA: IEEE, 2016, pp. 700–705.

[38] H. Idriss, T. Idriss, and M. Bayoumi, "A highly reliable dual-arbiter PUF for lightweight authentication protocols," in *2017 IEEE Int. Conf. RFID Technol. Appl. (RFID-TA)*, Warsaw, Poland: IEEE, 2017, pp. 248–253.

[39] M. Majzoobi, F. Koushanfar, and M. Potkonjak, "Testing techniques for hardware security," in *2008 IEEE Int. Test Conf.*, Santa Clara, CA, USA: IEEE, 2008, pp. 1–10.

[40] P. Sedcole and P. Y. K. Cheung, "Within-die delay variability in 90nm FPGAs and beyond," in *2006 IEEE Int. Conf. Field Programmable Technol.*, Bangkok, Thailand: IEEE, 2006, pp. 97–104.

[41] J. Delvaux, R. Peeters, D. Gu, and I. Verbauwhede, "A survey on lightweight entity authentication with strong PUFs," *ACM Comput. Surv.*, vol. 48, no. 2, p. 26, Oct. 2015, doi: 10.1145/2818186.

[42] G. T. Becker, "On the pitfalls of using arbiter-PUFs as building blocks," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 34, no. 8, pp. 1295–1307, Apr. 2015, doi: 10.1109/TCAD.2015.2427259.

[43] N. Hansen, "The CMA evolution strategy: A comparing review," in *Towards a New Evolutionary Computation: Advances in the Estimation of Distribution Algorithms*, J. A. Lozano, P. Larrañaga, I. Inza, and E. Bengoetxea, Eds., Berlin, Heidelberg: Springer, 2006, pp. 75–102.

[44] J. Delvaux, "Machine-learning attacks on PolyPUFs, OB-PUFs, RPUFs, LHS-PUFs, and PUF–FSMs," *IEEE Trans. Inf. Forensics Security*, vol. 14, no. 8, pp. 2043–2058, Jan. 2019, doi: 10.1109/TIFS.2019.2891223.

[45] G. Hospodar, R. Maes, and I. Verbauwhede, "Machine learning attacks on 65nm Arbiter PUFs: Accurate modeling poses strict bounds on usability," in *2012 IEEE Int. Workshop Inf. Forensics Secur. (WIFS)*, Costa Adeje, Spain: IEEE, 2012, pp. 37–42.

[46] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980," 2014.

[47] Z. Wen, J. Shi, Q. Li, B. He, and J. Chen, "ThunderSVM: A fast SVM library on GPUs and CPUs," *J. Mach. Learn. Res.*, vol. 19, no. 1, pp. 797–801, Jun. 2018.

AUTHOR BIOGRAPHIES

TAREK A. IDRISS received his B.Sc. degree in computer engineering from the University of Balamand, Lebanon, and his M.Sc. and Ph.D. degrees in Computer Engineering from the University of Louisiana at Lafayette, USA. Dr. Idriss is now an Assistant Professor at Western Washington University in Bellingham, WA, USA, where he teaches systems and security courses. His research interests include lightweight security, nanosatellites, and the Internet of Things.

HAYTHAM A. IDRISS received his B.Sc. in computer engineering from the University of Balamand, Lebanon, in 2009 and his M.Sc. in computer engineering from the University of Louisiana at Lafayette in 2016. He is currently a Ph.D. candidate in computer engineering at The Center for Advanced Computer Studies at the University of Louisiana at Lafayette. His research interests include fault tolerance design, hardware security, and RFID.

MAGDY A. BAYOUMI received his B.Sc. and M.Sc. degrees in electrical engineering from Cairo University, Egypt, his M.Sc. degree in computer engineering from Washington University in St. Louis, and his Ph.D. degree in electrical engineering from the University of Windsor, Ontario. Dr. Magdy A. Bayoumi is the director of The Center for Advanced Computer Studies (CACS) and department head of the Computer Science Department at the University of Louisiana at Lafayette, LA, USA. He is also the Z.L. Loflin Eminent Scholar Endowed Chair Professor in Computer Science. He was the Vice President for Conferences of the IEEE Circuits and Systems Society. He is a recipient of the 2009 IEEE Circuits and Systems Meritorious Service Award and the IEEE Circuits and Systems Society 2003 Education Award. His current research interests include VLSI designs and architectures, digital signal processing, and wireless ad hoc and sensor networks.