On-Demand Device Authentication using Zero-Knowledge Proofs for Smart Systems

Yadi Zhong
Auburn University
Auburn, Alabama, USA
yadi@auburn.edu

Joshua Hovanes
Auburn University
Auburn, Alabama, USA
josh.hovanes@auburn.edu

Ujjwal Guin
Auburn University
Auburn, Alabama, USA
ujjwal.guin@auburn.edu

ABSTRACT

Due to the exponential growth of IoT devices across diverse applications, it has become essential to secure edge devices against various hardware attacks, such as tampering and cloning. A tampered device with a hardware Trojan can bypass the security measures implemented through the software layers. One of the primary ways to verify the authenticity of a device is by using physically unclonable functions (PUFs) as a unique device fingerprint. During authentication, the PUF response from the edge device is transferred securely and compared with the stored response. This requires a secure communication setup between the edge device and the central server. The fingerprint must also be stored on a server for response matching. However, the potential compromise of the central server will result in the leak of all secret information of the edge devices, and adversaries can exploit it to gain unauthorized access to the IoT network. In this paper, we propose an efficient, secure, and on-demand communication protocol using zero-knowledge proofs (ZKPs) that allow the prover to provide evidence of its secret without revealing that to the verifier. The edge device, acting as the prover, convinces the central server, the verifier, of the unique PUF response stored inside the device without needing the actual storage of PUF responses on the server. The non-interactive characteristic of zk-SNARK, a widely used ZKP protocol in many popular cryptocurrencies such as Zcash, offers better optimization to authentication frequency, communication bandwidth between device and server, and protection of device-specific secret, all of which contribute to constructing our proposed device authentication framework.

CCS CONCEPTS

• **Security and privacy → Authentication**; **Embedded systems security**; **Hardware security implementation**.

KEYWORDS

Zero-knowledge proofs, zk-SNARK, physically unclonable functions, Internet of Things.

ACM Reference Format:

Yadi Zhong, Joshua Hovanes, and Ujjwal Guin. 2023. On-Demand Device Authentication using Zero-Knowledge Proofs for Smart Systems. In *Proceedings of the Great Lakes Symposium on VLSI 2023 (GLSVLSI '23), June 5–7, 2023, Knoxville, TN, USA.* GLSVLSI, Knoxville, TN, USA, 6 pages. https://doi.org/10.1145/3583781.3590275

GLSVLSI '23, June 5–7, 2023, Knoxville, TN, USA

ACM ISBN 979-8-4007-0125-2/23/06…$15.00
https://doi.org/10.1145/3583781.3590275

1 INTRODUCTION

Over the past decades, the world has experienced a booming Internet along with its wide range of web applications. The success of the interconnected computer network fuels the emergence of lightweight Internet of Things (IoTs), or edge devices. In turn, the advancement and adoption of the Internet of Things (IoT) and cyber-physical systems (CPS) offer foundations for smart cities and offered solutions to critical infrastructures such as transportation, public safety, health care, smart grid, etc. With 14.4 billion IoT devices currently connected to the Internet [2], this number is estimated to climb to 55.7 billion by 2025 [30]. These low-cost devices are deployed in the field for better integrating computing, cyber, and physical spaces, often in which proprietary information is included. However, this abundance of devices presents a severe challenge to ensuring the overall security of the connected infrastructure, as a genuine device can be replaced with a cloned one with a backdoor. Authenticating each device is a must to ensure the integrity and security of the IoT network as a whole. The insider threat [15, 21] is a looming issue, as it is challenging to monitor all edge devices while they are operational. An adversary can replace an authentic device with its cloned version and can gain access to the entire infrastructure, bypassing all security measures [7, 31, 35]. It is essential to authenticate an edge device regularly so the server can periodically evaluate and monitor its fidelity. The response received from an edge device serves as an indicator of its true identity.

Device authentication mechanisms using physically unclonable functions (PUFs) have been proposed over the years [3, 12, 13, 18, 32, 40]. Unfortunately, PUFs can be modeled if an adversary observes a set of challenge-response pairs (CRPs), and the responses can be predicted without accessing the physical device [4, 16, 22, 37, 38]. It is necessary to develop a novel communication protocol where an adversary cannot access PUF responses and thus cannot model the PUF. In addition, the secrecy of the PUF response should be kept secure even after repeated authentication. When deployed, the device must be checked regularly to detect the malicious replacement with a cloned counterpart. The same level of trust for a device must be maintained after post-deployment [29]

This paper proposes an efficient, secure, and on-demand device authentication protocol using zero-knowledge proofs (ZKPs) [26]. The signature from a physically unclonable function (PUF) is used to create the proof, which can be verified by the central server acting as a verifier without storing the actual PUF response. The edge device periodically generates proofs using a self-generated random seed. The proof can then be made public for verification for anyone with the common reference string (CRS). This allows repeated authentication by verifying the identity of an edge device without access to the PUF secret.

The contributions of this paper are summarized as follows:
• *Novel ZKP-based authentication protocol:* We propose an authentication scheme that uses ZKP to prove the existence of a device fingerprint. The PUF responses are kept within the device itself without sharing the secret value with the central server. Consequently, the server does not need to keep the PUF responses for every edge device operating in the field. We only require it to save the error correction syndromes for corresponding challenges in the server. The PUF response is processed inside the edge device, which is the source for generating zero-knowledge proofs. The server verifies the proof in combination with the publicly available parameters determined at the setup phase. The proof verification is computationally lightweight, and the server can process a batch of verification requests simultaneously. In addition, our method can leverage the non-interactive nature of zk-SNARK to allow any party access to the public CRS and proof to verify its legitimacy. This can potentially ease the workload of the central server in the presence of many authentication queries, where a server can outsource a few verification jobs to trusted delegates.
• *Resilience against machine learning-based attacks:* Our proposed zk-SNARK-based communication protocol does not require storing the PUF challenge-response pairs (CRP) on the central server. Instead, the CRS and the hashed values of the PUF response are stored for each edge device. The zero-knowledge property of zk-SNARKs ensures the verification of the PUF responses without revealing this secret to the verifier. This significantly improves the overall security of the IoT infrastructure, as the fingerprint of all edge devices will be protected even if the central server is compromised. In case of a suspected compromise of a PUF response, we can securely update the public statement itself by choosing a different challenge instead. After the challenge is updated, the edge device only needs to provide the updated statement to the server as the incurred communication overhead. As the adversary does not have access to the CRPs, it is not feasible to model the PUF, and thus our proposed approach is resilient against machine learning-based attacks.

The rest of the paper is organized as follows. We introduce PUF and ZKP in Section 2. The proposed IoT device authentication framework is described in Section 3. The experimental result and performance analysis for the proposed approach are described in Section 4. Finally, we conclude the paper in Section 5.

2 BACKGROUND

This section presents an overview of physically unclonable functions (PUF), the PUF-based secure communication protocols, and their challenges. We describe the prior work on ZKPs, Zero-Knowledge Succinct Non-Interactive Argument of Knowledge (zk-SNARK), and elliptic curve pairing-based zk-SNARK.

2.1 Physically Unclonable Functions

Throughout the years, the research community has proposed various architectures for physically unclonable functions (PUFs) and their implementations [23, 27, 28, 40]. PUFs derive their behavior from the inherently uncontrollable and unpredictable variations in the semiconductor manufacturing process. Therefore, its application involves security and privacy-related research for its ability to generate unclonable bits as a unique key for identification and authentication. Over the years, different types of PUFs have been proposed, *e.g.,* arbiter PUF [23], ring oscillator PUF (ROPUF) [40, 42], SRAM PUF [27, 28], among others. These designs leverage the unclonable characteristic in path delay and/or threshold voltage biases to demonstrate the uniqueness of PUF.

Multiple PUF-based device authentication protocols have been proposed in recent years [3, 12, 13, 18, 32, 40]. Arbiter PUFs and ROPUFs were first applied in device authentication [40], where the server keeps a database of CRPs for each PUF instance. SRAM PUF-based authentication protocols [3, 32] also explicitly store PUF responses on the server side for verification or key derivation. Chatterjee et al. does not keep PUF responses in the verifier's database, but the offlined security credential generator has those copies from the enrollment phase [12]. Although security is guaranteed for the above-mentioned algorithms with unclonable PUFs, machine learning attacks with a subset of given challenge-response pairs (CRPs) have been demonstrated effective in accurately predicting responses [4, 16, 22, 37, 38]. Rührmair et al. first exploits logic regression for learning the features in CRPs to predict unseen Arbiter and XOR Arbiter PUF responses with high probability [37]. Subsequently, different machine learning-based attacks have surfaced to target the extrapolation of PUF responses from various PUF architectures [4, 16, 22, 38].

2.2 Zero-Knowledge Proofs

The overarching goal of ZKPs is to convince a verifier that a prover possesses an existing secret without ever revealing the secret. These proofs were conceptualized initially in 1985 by Goldwasser et al. [25]. This laid the foundation for the subsequent development of the Feige-Fiat-Sharir protocol, one of the first identification schemes using interactive ZKPs [19]. In interactive ZKPs, real-time communication between the prover and the verifier is necessary to complete the protocol. Non-interactive ZKPs, on the other hand, relieve both parties from repeated communications. This concept is first explored through Fiat-Shamir heuristic [20] and then developed by Blum et al. [8]. The non-interactive nature and the adopted random oracle model allow any entity to verify the validity of a proof, where the proof generation no longer relies on the verifier-dependent random values. Succinct non-interactive zero-knowledge proofs (zk-SNARKs) produce logarithmic size proofs for verification [5]. The popular cryptocurrency ZCash uses the Zerocash framework [39] with zk-SNARK proof system to ensure the anonymity of user identity [34]. Over the years, different zk-SNARKs protocols have been proposed and quickly gained popularity in privacy-based blockchain applications, such as Pinocchio [36], Groth'16 [26], Bulletproofs [10], Sonic [33], Marlin [14], etc.

2.3 Pairing-based zk-SNARK [26]

Despite various ZKPs having been proposed recently, the zk-SNARK proposed by Groth [26] remains to be very efficient both for the shorter proof size and less computation complexity for the verifier [26, 33]. Groth'16 operates on pairing-based cryptography, whose underlying cryptographic hardness lies in the bilinear Diffie-Hellman problem on elliptic curves [9]. It constructs the rank-1 constraint system (R1CS) of a quadratic arithmetic program (QAP) with ℓ public statement, where each gate represents a multiplication/exponentiation operation on elliptic curve. Computations are performed under bilinear groups $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ that forms a non-degenerate bilinary mapping $\mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$. The constant proof size contains 3 group elements, 2 in $\mathbb{G}_1$ and 1 in $\mathbb{G}_2$. Once a proof is

received, the verifier will compute ℓ exponentiation in $\mathbb{G}_1$, 3 pairing values, and check the quantitative equivalence of pairings in $\mathbb{G}_T$.

3 PROPOSED APPROACH

This section details our proposed zk-SNARK-based approach for IoT device authentication. The zero-knowledge scheme provides security to PUF-based IoT verification methods without revealing the secret asset in each edge device, *e.g.,* PUF response. Figure 1 shows the overview of the proposed method, which consists of two phases, the registration phase and the operational phase. The first phase consists of PUF calibration and response extraction, and initialization of parameters necessary for the trusted setup of zk-SNARK. This phase is performed in a trusted environment right before the deployment of IoT devices. Once the device is in the field, it can be periodically checked and verified by the trusted server by validating zk-SNARK proofs it generated.

3.1 Threat Model

This paper assumes the adversarial capability of replacing the authentic device with a counterfeit, tampered, or cloned device. In addition, the threat model assumes that the device does not provide direct access of the PUF challenge-response pairs. This is practical as the semiconductor industry typically disables the scan chain access after the manufacturing test [11]. Similar features of blocking scan access can be implemented here to prevent attackers from obtaining the PUF responses. The adversary simply cannot bypass the security mechanism while the device is in the field.

3.2 Registration Phase

Registration of the edge device begins with creating a stable PUF response at a given challenge. We follow the pairing-based zk-SNARK construction of Groth'16 [26] to authenticate and verify IoT devices using each device's inherent secret asset without revealing it to them. In the registration phase, both the central server and the edge device need to build a mutually agreed relation $\mathcal{R}$ and trusted setup σ, as shown in steps ① and ② in Figure 1. Additionally, a valid hash for the PUF response and its error correction syndrome will be computed and made public by the edge device, as shown in steps ③ and ④ in Figure 1. This initialization phase can be completed in a secure environment as the central server is considered as trusted.

• **Construct Relation $\mathcal{R}$:** Before constructing and validating proofs, both parties need to agree on the public parameters, similar to Alice and Bob coordinating on the same prime and primitive root for computations in the Diffie-Hellman protocol [17]. In the ZKP protocol, the relation $\mathcal{R}$ is mutually communicated between both the prover and the verifier. Let $\mathcal{R}$ be a relation of a given security parameter λ with prime field $\mathbb{F}_p$, generators $g_1, g_2 \in \mathbb{F}_p$ of elliptic curve groups $\mathbb{G}_1, \mathbb{G}_2$ of prime order q, bilinear mapping $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ of prime order q with generator $g_t = e(g_1, g_2)$, scalars $m, \ell, 3m$ polynomials $\{f_{A_i}(X), f_{B_i}(X), f_{C_i}(X)\}_{i=1}^{m}$ of degree $n-1$, and polynomial $f_Z(X)$ of degree n,

$$\mathcal{R} = \left(\begin{array}{c} p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, \ell, \\ \{f_{A_i}(X), f_{B_i}(X), f_{C_i}(X)\}_{i=1}^{m}, f_Z(X) \end{array} \right).$$

Due to the process variations in chip manufacturing, the SRAM inside each IoT device contains its own unique signature, *i.e.,* the power-up state of the SRAM cells [27, 41]. It is feasible to extract an SRAM PUF from the stable bits by calibrating and eliminating the unstable bits from the power-up state. This is applicable to other types of PUFs also. A set of challenges $\{c_1, \ldots, c_h, \ldots\}$ can be constructed for a PUF. For a given challenge c_h, one can obtain a unique PUF response w_h for each device and the corresponding error correction syndrome y_h to ensure consistency in w_h. Due to the unclonable property of the PUF response, the IoT device has secret witness w_h, and the corresponding hash value as the public statement $s_h = \mathsf{hash}(w_h)$. The concatenation of witness w_h and statement s_h is the solution to the quadratic arithmetic circuit in R1CS defined over $\{f_{A_i}(X), f_{B_i}(X), f_{C_i}(X)\}_{i=1}^{m}$ and $f_Z(X)$,

$$\sum_{i=1}^{m} d_i f_{A_i}(X) \cdot \sum_{i=1}^{m} d_i f_{B_i}(X) - \sum_{i=1}^{m} d_i f_{C_i}(X) = f_Q(X) f_Z(X),$$

where $f_Q(X)$ is the quotient polynomial of degree $n-2$, $s_h = (d_1, \ldots, d_\ell) \in \mathbb{F}_p^{\ell}$, $w_h = (d_{l+1}, \ldots, d_m) \in \mathbb{F}_p^{m-\ell}$, and $d = (d_1, \ldots, d_m) = (s_h, w_h) \in \mathbb{F}_p^{m}$. As it involves the secret witness w_h, the quotient polynomial $f_Q(X)$ is computed by the edge device and is not shared with the central server.

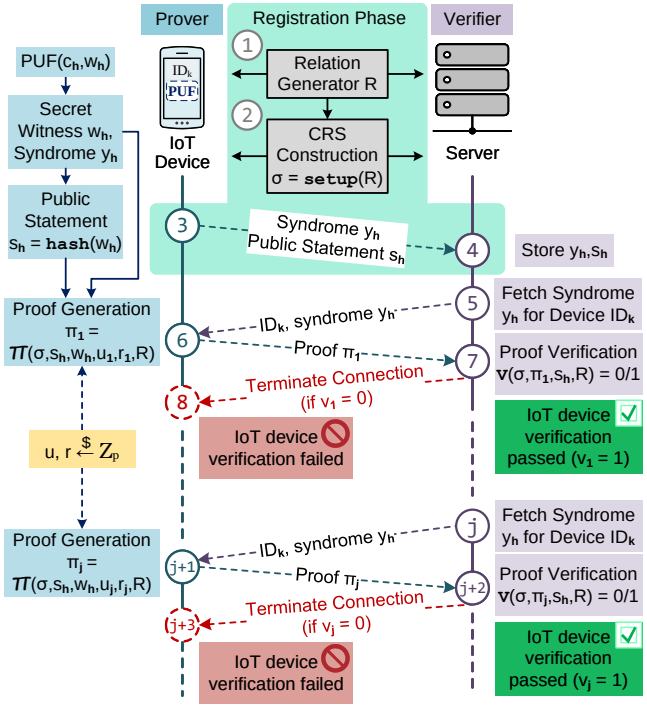

Figure 1: The proposed device authentication scheme using ZKPs.

• **Initialize Trusted Setup σ:** During the trusted setup, the IoT device and the central server compute the common reference string (CRS), which is essentially the basis for both proof generation and validation. It applies the polynomials derived from the relation $\mathcal{R}$ with R1CS arithmetic circuit to construct the public elliptic curve points in $\mathbb{G}_1, \mathbb{G}_2$. It is called a trusted setup due to the fact that the creation of CRS σ needs the uniform sampling of 5 random values $\alpha, \beta, \gamma, \delta, x$ from the prime field $\mathbb{F}_p$, denoted as $\alpha, \beta, \gamma, \delta, x \xleftarrow{\$} \mathbb{F}_p$. Note that, in the registration phase, we consider both the edge device and the servers as trusted. One can select these variables at random and from the uniform distribution. Once these values are determined, elements in both cyclic groups $\mathbb{G}_1, \mathbb{G}_2$ are generated to construct the CRS σ,

$$\sigma = \left(\begin{array}{c} g_1^{\alpha}, g_1^{\beta}, g_1^{\delta}, \{g_1^{x^i}\}_{i=0}^{n-1}, \{g_1^{\delta^{-1}x^i f_Z(x)}\}_{i=1}^{n-2} \\ \left\{g_1^{\gamma^{-1}\cdot\left(\beta f_{A_i}(x)+\alpha f_{B_i}(x)+f_{C_i}(x)\right)}\right\}_{i=1}^{\ell} \\ \left\{g_1^{\delta^{-1}\cdot\left(\beta f_{A_i}(x)+\alpha f_{B_i}(x)+f_{C_i}(x)\right)}\right\}_{i=\ell+1}^{m} \\ g_2^{\beta}, g_2^{\gamma}, g_2^{\delta}, \{g_2^{x^i}\}_{i=0}^{n-1}, g_T^{\alpha\beta} \end{array} \right).$$

The pairing $g_T^{\alpha\beta} = e(g_1^{\alpha}, g_2^{\beta})$ is also evaluated to improve further the computational succinctness for the verifier (server). Note that this device-specific CRS is stored on both parties and can be made public. As for the server, it is required to distinguish the CRS from one edge device to another, where a unique set of parameters in σ is constructed from each IoT device with its secret witness.

• **Generate and Publicize Syndrome y_h and Statement s_h:** Although error correction syndrome is required for computing the stable PUF response, each edge device can further optimize its resource utilization without saving them on the device itself. The syndromes for all challenges in set $\{c_1, \ldots, c_h, \ldots\}$ can be sent to the server instead. Syndrome y_h is computed over the PUF response w_h given the challenge c_h, where the server will transmit it back to the device before proof generation. Proving the knowledge of a secret witness, *i.e.,* PUF response, involves statements in non-deterministic polynomial time complexity (NP) [25]. It can be performed by proving to the verifier the knowledge of the preimage of a hash generated from collision-resistant and preimage-resistant hash algorithm [6]. This requires the server to be able to check the statement of a valid confirmed response against a newly generated one. Thus, during the trusted phase, the statement $s_h = \mathsf{hash}(w_h)$ is computed at the IoT node and is then made public by the corresponding node. The device can simply broadcast its s_h value in the IoT network. Steps ③ and ④ of Figure 1 show the transmission and acceptance of s_h from the device to the server. Note that the adversary cannot obtain the PUF response from its hash s_h due to the preimage resistance of the secure hash algorithm. In case of updating the PUF response with another challenge $c_{h'}$, new statement and syndrome can be generated and made public based on the updated response $w_{h'}$.

3.3 Operational Phase

Once the IoT device is deployed in the field, the central server needs to ensure that adversaries cannot masquerade a malicious device as the genuine one, nor can they replace an authentic device with a clone/counterfeit one. Therefore, it is important to guarantee the authenticity of the device, where the server needs to query the device for a response and evaluate or certify the legitimacy of the device. To counter the attacker from the malicious substitution of edge devices, we propose the periodic authentication of devices, where proofs are generated from IoT nodes and are delivered to the server for verification. For each device, we denote its j^{th} proof π_j and the corresponding verification v_j with subscript j, where $j = 1, 2, 3, \ldots$ The proposed on-demand authentication allows the prover to initiate the proof generation, independent from the verifier, the server. In case of the required error correction syndrome needs to be retrieved online, the server can then initiates the proof-verification process by sending the corresponding syndrome y_h to the edge device for error correction in PUF extraction, as in step ⑤ in Figure 1. Proof generation begins in an IoT device after it obtains the witness w_h with challenge c_h and syndrome y_h.

• **Proof Generation π_j:** To prevent an adversary's malicious replacement of the edge device while deployed in the field, we secure the device authentication with proof generation and validation from the zk-SNARK protocol. The authentic device is trying to prove to the server that it knows the preimage of the hash value s_h without disclosing it to the server. The proofs in the zk-SNARK reveal nothing about the secret witness itself, *i.e.,* perfect zero-knowledge, due to the fact that it incorporates additional random values. In order to generate multiple proofs for the same device at different time intervals, it can be done for the j^{th} proof π_j by randomly selecting a different set of (u, r), sampled from $\mathbb{F}_p$ with uniform distribution, *e.g.,* a random oracle, denoted as $u_j, r_j \xleftarrow{\$} \mathbb{F}_p$. Both the secret witness $w_h = (d_{l+1}, \ldots, d_m)$ and the public statement $s_h = (d_1, \ldots, d_\ell)$ are used during the proof construction. The proof $\pi_j = \Pi(\sigma, s_h, w_h, u_j, r_j, \mathcal{R}) = (\pi_A, \pi_B, \pi_C)$ consists of three elliptic curve points (π_A, π_B, π_C) with $\pi_A, \pi_C \in \mathbb{G}_1$ and $\pi_B \in \mathbb{G}_2$,

$$\pi_j = \left(\; \pi_A = g_1^{z_1},\ \pi_B = g_2^{z_2},\ \pi_C = g_1^{r_j z_1 + u_j z_2 + z_3 - u_j r_j \delta} \; \right),$$

where z_1, z_2, z_3 are computed as follows.

$$z_1 = \alpha + \sum_{i=1}^{m} d_i f_{A_i}(x) + u_j\delta;\ z_2 = \beta + \sum_{i=1}^{m} d_i f_{B_i}(x) + r_j\delta;$$

$$z_3 = \delta^{-1} \sum_{i=\ell+1}^{m} d_i\left(\beta f_{A_i}(x) + \alpha f_{B_i}(x) + f_{C_i}(x)\right) + f_Q(x) f_Z(x)$$

• **Proof Verification v_j:** Once proof π_j has been received, the verifier can check the validity of the device by evaluating π_j with the public statement s_h. The server performs computations in $\mathbb{G}_T$ due to the bilinear property of $e(g_1^a, g_2^b) = e(g_1, g_2)^{ab}$ in $\mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$. The verification $v_j = \mathcal{V}(\sigma, \pi_j, s_h, \mathcal{R})$ check whether the following equality holds,

$$e(\pi_A, \pi_B) = g_t^{\alpha\beta} \cdot e(g_1^{\phi}, g_2^{\gamma}) \cdot e(\pi_C, g_2^{\delta})$$

with ϕ defined as:

$$\phi = \gamma^{-1}\Big(\sum_{i=1}^{\ell} d_i\left(\beta f_{A_i}(x) + \alpha f_{B_i}(x) + f_{C_i}(x)\right)\Big).$$

The pairing equality is satisfied only when the prover, the IoT node, has full knowledge of the input w_h that matches $s_h = \mathsf{hash}(w_h)$. This is due to the negligible probability of the adversary succeeding in convincing the verifier of a false statement or finding the preimage of hash s_h, *e.g.,* the soundness property of zero-knowledge argument. If the device verification passes ($v_j = 1$), the device is considered trusted at present, and communications and interactions can proceed as normal. If the server cannot validate the proof ($v_j = 0$), the device authentication fails, and that IoT node is marked as untrusted. The server can then terminate its connection to the specific edge device, followed by additional countermeasures if possible.

$$v_j = \begin{cases} 1, & e(\pi_A, \pi_B) = g_t^{\alpha\beta} \cdot e(g_1^{\phi}, g_2^{\gamma}) \cdot e(\pi_C, g_2^{\delta}) \\ 0, & e(\pi_A, \pi_B) \neq g_t^{\alpha\beta} \cdot e(g_1^{\phi}, g_2^{\gamma}) \cdot e(\pi_C, g_2^{\delta}) \end{cases}$$

With a predefined interval set between the server and each edge device, the IoT node can periodically send proofs to server and get it verified. For example, step ⑥ of Figure 1 shows device generates its first proof π_1 with random values $(u_1, r_1) \xleftarrow{\$} \mathbb{F}_p$. The server verifies proof π_1 in step ⑦, and it would issue termination of connection if proof verification failed ($v_1 = 0$). Similarly, the device generates the j^{th} proof π_j at the appointed time interval with

syndrome y_h (step (j)) after all prior $(j-1)$ proofs are validated, shown in step (j+1) of Figure 1. The server would perform the same verification as before to determine the authenticity of the device.

3.4 Security Analysis

The zk-SNARK protocol offers the proposed device authentication framework with additional security strength over the conventional counterparts. We analyze and provide proof sketch for the security strength of the proposed framework under three fundamental properties of ZKP, completeness, soundness, and zero-knowledge [25].

• **Machine Learning Attack Resistance Analysis:** As the device fingerprint is proved to the verifier without transmitting it or revealing it through the communication channel, the adversary will not find the secret fingerprint when monitoring the data packets between the device and server. The zero-knowledge property guarantees the proof π_j reveals nothing about the secret witness itself or the PUF response w_h. Moreover, due to the preimage resistance of the hash algorithm, the probability of the attacker's success in finding the preimage w_h from its hash value s_h is low and generally less likely than the probability of hash collision. For the widely adopted MiMC hash construction [1], it has a 256-bits preimage security and a 128-bits collision security. This means the adversary cannot determine the PUF response of a device from its hash. As no response is ever leaked to the attacker during the IoT device authentication, it is not practical to construct a set of known CRPs for training the machine learning algorithm nor the subsequent attack to predict the unseen PUF response. Thus, it is infeasible for a malicious actor to launch machine learning-based attacks to model the PUF, as no training dataset can be extracted.

• **Proof Verification with Trusted Delegates:** Conventional device authentication schemes have a designated machine (*i.e.,* server) for validating the IoT device. This is due to the fact that the challenge-response pairs of edge devices, along with the public error correction syndromes, are stored in the secure server. Recall that the CRS σ and statement s_h are treated as public parameters, the server can essentially outsource the verification to some trusted parties for performing the necessary computation instead. With the soundness property, a non-polynomial time adversary would be impossible to use randomly generated proofs to convince the trusted delegates (or the server) of the validity of fake proof [26]. For honest prover, the completeness property ensures that its proof can be validated by verification protocol. Therefore, having trusted delegates, it is possible to ease the workload of the central server in response to huge verification queries.

• **Replay Attack Prevention:** The proof validation relies on matching pairing-based computation of hash value and proof. However, considering these proofs are made public after generation, an adversary that passively observes the communication channel could save these valid proofs and reuse them if the edge device was compromised. To prevent this, an extra layer of security must be added to save and check if a proof has been used before, as newly generated proofs will be unique. Therefore, the server may need to save each proof for later comparison to check for proof uniqueness.

4 EXPERIMENTAL RESULTS

We have implemented our proposed IoT device authentication protocol with the Raspberry Pi 4 Model B. This device is commonly used in IoT applications today and is one of the most flexible microcontrollers on the market. Proof verification is done on a Ubuntu laptop, acting as the central server. Our proposed authentication protocol is built on top of the ZPiE library [24] with Groth'16 zk-SNARK [26]. Figure 2 shows the setup of our implementation, where edge devices and the central server communicate through a wireless network. The edge device creates a proof once it periodically receives the server's request for its authentication.

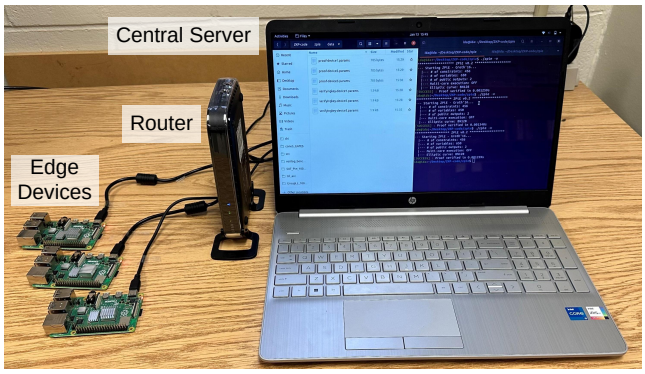

Figure 2: Implementation setup with Raspberry Pi 4 Model B as edge devices and laptop as the central server.

4.1 Performance Analysis

Although zk-SNARK proof generation has a relatively high computational workload than proof verification, it only needs to be performed sparingly to be used effectively. Note that the zk-SNARK is designed to allow the prover to convince the verifier of the knowledge of its secret in a single proof, due to the negligible success of a dishonest prover cheating with a false statement under a sound zk-SNARK protocol [26]. For edge devices already deployed in the field, monitoring with regular authentication is needed to ensure the integrity of the device. In our proposed authentication scheme, the proof generation is performed on device startup to simulate a periodic yet moderately frequent device verification schedule. As the device would likely be powered off during any form of replacement, this schedule would check at each possible window while not using extremely high amounts of computational power. Figure 3 shows the distributions of both proof generation and verification with box plots. Over 60 tests on the Raspberry Pi 4 Model B using ZPiE's single-threaded setting, the average proof generation time is 232.7 ms. Similarly, the average verification time of Groth'16 proofs on the server is 1.54 ms. This advantage of fast proof verification allows the server to compute them frequently without noticeable delays in simultaneously fulfilling verification requests from multiple devices. The computationally lightweight proof verification by the server permits the rapid expansion of enrolled devices in the IoT network as the verification time is scalable to a larger IoT device count. Overall, the verification time is extremely short, and the proving time is also short when considering the scale and frequency with which this protocol would be performed.

5 CONCLUSION

This paper presents a novel authentication protocol for IoT devices using the state-of-the-art zk-SNARK algorithm with SRAM PUF as the device fingerprint. Our proposed ZKP-based approach can guarantee the secrecy of the PUF response even if the central server

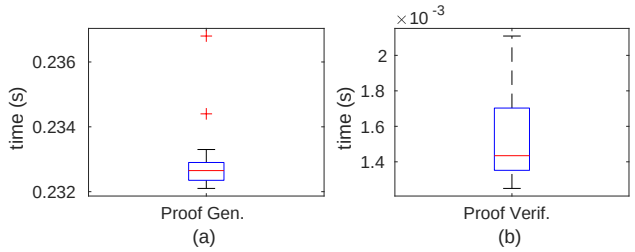

Figure 3: Box plots for (a) proof generation time by Raspberry Pi acting as IoT devices, (b) proof verification time by a laptop as a server.

gets compromised. It resists machine learning-based attacks and the attacker's attempts to extrapolate the PUF responses, as the adversary can never observe any challenge-response pair from analyzing the entire communication between the server and a particular edge device. Our proposed scheme can further prevent the potential replay attack or impersonation if the adversary monitors the communication channel. In addition, delegating the verification work to trusted parties is possible when the server is busy with different verification requests. We demonstrated our proposed framework by implementing it on Raspberry Pi 4 Model B boards as edge devices with ZPiE open-source library. Proofs generated by the IoT devices are sent to the server via a wireless network, and the server can verify them very efficiently within a few milliseconds.

REFERENCES

[1] Martin Albrecht, Lorenzo Grassi, Christian Rechberger, Arnab Roy, and Tyge Tiessen. 2016. MiMC: Efficient Encryption and Cryptographic Hashing with Minimal Multiplicative Complexity. In *Int. Conf. on the Theory and Application of Cryptology and Information Security*. Springer, 191–219.

[2] IoT Analytics. 2022. *State of IoT 2022: Number of connected IoT devices growing 18% to 14.4 billion globally.* https://iot-analytics.com/number-connected-iot-devices/

[3] Aydin Aysu, Ege Gulcan, Daisuke Moriyama, Patrick Schaumont, and Moti Yung. 2015. End-to-end design of a PUF-based privacy preserving authentication protocol. In *Cryptographic Hardware and Embedded Systems (CHES)*. 556–576.

[4] Georg T Becker. 2015. The gap between promise and reality: On the insecurity of XOR arbiter PUFs. In *Cryptographic Hardware and Embedded Systems (CHES)*. 535–555.

[5] Eli Ben-Sasson, Alessandro Chiesa, Eran Tromer, and Madars Virza. 2014. Succinct {Non-Interactive} Zero Knowledge for a von Neumann Architecture. In *23rd USENIX Security Symposium (USENIX Security 14)*. 781–796.

[6] Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Eran Tromer. 2012. From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*. 326–349.

[7] Bloomberg. 2018. *The Big Hack: How China Used a Tiny Chip to Infiltrate U.S. Companies.* https://www.bloomberg.com/news/features/2018-10-04/the-big-hack-how-china-used-a-tiny-chip-to-infiltrate-america-s-top-companies

[8] Manuel Blum, Paul Feldman, and Silvio Micali. 1988. Non-Interactive Zero-Knowledge and Its Applications. In *Symp. on Theory of Computing*. 103–112.

[9] Dan Boneh and Matt Franklin. 2001. Identity-based encryption from the Weil pairing. In *Int. Cryptology Conference*. Springer, 213–229.

[10] Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Greg Maxwell. 2018. Bulletproofs: Short proofs for confidential transactions and more. In *IEEE Symp. on Security and Privacy (SP)*. 315–334.

[11] Michael Bushnell and Vishwani Agrawal. 2004. *Essentials of Electronic Testing for Digital, Memory and Mixed-Signal VLSI Circuits*. Vol. 17. Springer Science & Business Media.

[12] Urbi Chatterjee, Vidya Govindan, Rajat Sadhukhan, Debdeep Mukhopadhyay, Rajat Subhra Chakraborty, Debashis Mahata, and Mukesh M Prabhu. 2018. Building PUF based authentication and key exchange protocol for IoT without explicit CRPs in verifier database. *IEEE Trans. on Dependable and Secure Computing* 16, 3 (2018), 424–437.

[13] Wenjie Che, Fareena Saqib, and Jim Plusquellic. 2015. PUF-based authentication. In *International Conference on Computer-Aided Design (ICCAD)*. 337–344.

[14] Alessandro Chiesa, Yuncong Hu, Mary Maller, Pratyush Mishra, Noah Vesely, and Nicholas Ward. 2020. Marlin: preprocessing zkSNARKs with universal and updatable SRS. In *International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 738–768.

[15] Cybersecurity and Infrastructure Security Agency (CISA). 2022. *Insider Threat Mitigation.* https://www.cisa.gov/insider-threat-mitigation

[16] Jeroen Delvaux. 2019. Machine-Learning Attacks on PolyPUFs, OB-PUFs, RPUFs, LHS-PUFs, and PUF–FSMs. *IEEE Transactions on Information Forensics and Security* 14, 8 (2019), 2043–2058.

[17] Whitfield Diffie and Martin E Hellman. 1976. New Directions in Cryptography. *IEEE Transactions on Information Theory* 22, 6 (1976).

[18] Mohammed El-Hajj, Ahmad Fadlallah, Maroun Chamoun, and Ahmed Serhrouchni. 2019. A Survey of Internet of Things (IoT) Authentication Schemes. *Sensors* 19, 5 (2019), 1141.

[19] Uriel Feige, Amos Fiat, and Adi Shamir. 1988. Zero-knowledge proofs of identity. *Journal of cryptology* 1, 2 (1988), 77–94.

[20] Amos Fiat and Adi Shamir. 1986. How to Prove Yourself: Practical Solutions to Identification and Signature Problems.. In *Crypto*, Vol. 86. Springer, 186–194.

[21] Forbes. 2023. *Was The Cybersecurity Crystal Ball Cloudy Or Clear Two Years Ago?*

[22] Fatemeh Ganji, Shahin Tajik, Fabian Fäßler, and Jean-Pierre Seifert. 2016. Strong machine learning attack against PUFs with no mathematical model. In *Cryptographic Hardware and Embedded Systems (CHES)*. 391–411.

[23] Blaise Gassend, Dwaine Clarke, Marten Van Dijk, and Srinivas Devadas. 2002. Silicon Physical Random Functions. In *ACM Conf. on Computer and Communications Security*. 148–160.

[24] GitHub. 2021. *ZPiE: Zero-knowledge Proofs in Embedded systems.* https://github.com/xevisalle/zpie

[25] Shafi Goldwasser, Silvio Micali, and Chales Rackoff. 1985. The Knowledge Complexity of Interactive Proof-Systems. In *Proceedings of the seventeenth annual ACM symposium on Theory of computing*. 291–304.

[26] Jens Groth. 2016. On the size of pairing-based non-interactive arguments. In *Int. conf. on theory and applications of cryptographic techniques*. Springer, 305–326.

[27] Jorge Guajardo, Sandeep S Kumar, Geert-Jan Schrijen, and Pim Tuyls. 2007. FPGA Intrinsic PUFs and Their Use for IP Protection. In *International workshop on Cryptographic Hardware and Embedded Systems*. Springer.

[28] Daniel E Holcomb, Wayne P Burleson, and Kevin Fu. 2008. Power-up SRAM state as an identifying fingerprint and source of true random numbers. *IEEE Trans. Comput.* 58, 9 (2008), 1198–1210.

[29] Joshua Hovanes, Yadi Zhong, and Ujjwal Guin. 2022. Beware of Discarding Used SRAMs: Information is Stored Permanently. In *IEEE Physical Assurance and Inspection of Electronics (PAINE)*. 1–7.

[30] International Data Corporation (IDC). 2021. *Future of Industry Ecosystems: Shared Data and Insights.* https://blogs.idc.com/2021/01/06/future-of-industry-ecosystems-shared-data-and-insights/

[31] Nima Karimian, Mark Tehranipoor, Damon Woodard, and Domenic Forte. 2019. Unlock your heart: Next generation biometric in resource-constrained healthcare systems and IoT. *IEEE Access* 7 (2019), 49135–49149.

[32] Jubayer Mahmod and Ujjwal Guin. 2020. A Robust, Low-Cost and Secure Authentication Scheme for IoT Applications. *Cryptography* 4, 1 (2020), 8.

[33] Mary Maller, Sean Bowe, Markulf Kohlweiss, and Sarah Meiklejohn. 2019. Sonic: Zero-knowledge SNARKs from linear-size universal and updatable structured reference strings. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*. 2111–2128.

[34] Ian Miers, Christina Garman, Matthew Green, and Aviel D. Rubin. 2013. Zerocoin: Anonymous Distributed E-Cash from Bitcoin. In *IEEE Symp. on Security and Privacy*. 397–411.

[35] NPR. 2014. *U.S. HVAC Firm Reportedly Linked To Target's Data Security Breach.* https://www.npr.org/sections/thetwo-way/2014/02/05/272101928/u-s-hvac-firm-reportedly-linked-to-target-s-data-security-breach

[36] Bryan Parno, Jon Howell, Craig Gentry, and Mariana Raykova. 2016. Pinocchio: Nearly Practical Verifiable Computation. *Commun. ACM* 59, 2 (2016), 103–112.

[37] Ulrich Rührmair, Frank Sehnke, Jan Sölter, Gideon Dror, Srinivas Devadas, and Jürgen Schmidhuber. 2010. Modeling attacks on physical unclonable functions. In *Proceedings of ACM Conf. on Computer and Communications Security*. 237–249.

[38] Ulrich Rührmair, Jan Sölter, Frank Sehnke, Xiaolin Xu, Ahmed Mahmoud, Vera Stoyanova, Gideon Dror, Jürgen Schmidhuber, Wayne Burleson, and Srinivas Devadas. 2013. PUF modeling attacks on simulated and silicon data. *IEEE transactions on information forensics and security* 8, 11 (2013), 1876–1891.

[39] Eli Ben Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. 2014. Zerocash: Decentralized anonymous payments from bitcoin. In *Symp. on Security and Privacy*. 459–474.

[40] G. Edward Suh and Srinivas Devadas. 2007. Physical Unclonable Functions for Device Authentication and Secret Key Generation. In *Design Automation Conference*. 9–14.

[41] Kan Xiao, Md Tauhidur Rahman, Domenic Forte, Yu Huang, Mei Su, and Mohammad Tehranipoor. 2014. Bit selection algorithm suitable for high-volume production of SRAM-PUF. In *IEEE Int. Symp. on Hardware-Oriented Security and Trust (HOST)*. 101–106.

[42] Xin Xin, Jens-Peter Kaps, and Kris Gaj. 2011. A configurable ring-oscillator-based PUF for Xilinx FPGAs. In *IEEE Euromicro conf. on digital system design*. 651–657.