AuthentiSafe: Lightweight and Future-Proof Device-to-Device Authentication for IoT

Lukas Petzi
University of Würzburg
Würzburg, Germany
lukas.petzi@uni-wuerzburg.de

Torsten Krauß
University of Würzburg
Würzburg, Germany
torsten.krauss@uni-wuerzburg.de

Alexandra Dmitrienko
University of Würzburg
Würzburg, Germany
alexandra.dmitrienko@uni-wuerzburg.de

Gene Tsudik
UC Irvine
Irvine, USA
gts@ics.uci.edu

Abstract

The Internet of Things (IoT) is permeating many aspects of everyday life, including homes, public spaces, and industrial settings. The growing popularity of IoT devices prompts an important challenge of ensuring secure and reliable communication. To support direct device-to-device communication among resource-constrained IoT devices, lightweight authentication techniques are needed that do not rely on trusted servers. One promising primitive for this purpose are Physical Unclonable Functions (PUFs), which utilize unique characteristics of hardware components (e.g., memory) inherent in the manufacturing of IoT devices. Given some input, a PUF generates a unique secret value which facilitates secure and robust authentication. However, current PUF-based authentication schemes require a trusted server, incur substantial computational costs, or lack resilience against emerging quantum computing threats.

This paper introduces AuthentiSafe, a lightweight, scalable, and secure PUF-based authentication scheme for device-to-device authentication among mutually mistrusting IoT devices. AuthentiSafe integrates PUFs with one-time signatures and cryptographic accumulators, thus eliminating the need for a trusted server. By relying exclusively on efficient cryptographic one-way hash functions, AuthentiSafe minimizes protocol costs and accommodates low computational power and very limited secure storage of IoT devices. It also ensures security in the post-quantum era, since one-way functions remain resilient to quantum attacks. We show AuthentiSafe's resilience against various attacks. Experiments show that it appreciably outperforms three prior PUF-based authentication schemes.

CCS Concepts

• **Security and privacy → Hardware-based security protocols**.

Keywords

PUF, IoT Authentication, IoT Security, Embedded Security

ASIA CCS '25, Hanoi, Vietnam

ACM ISBN 979-8-4007-1410-8/25/08
https://doi.org/10.1145/3708821.3710831

ACM Reference Format:
Lukas Petzi, Torsten Krauß, Alexandra Dmitrienko, and Gene Tsudik. 2025. AuthentiSafe: Lightweight and Future-Proof Device-to-Device Authentication for IoT. In *ACM Asia Conference on Computer and Communications Security (ASIA CCS '25), August 25–29, 2025, Hanoi, Vietnam.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3708821.3710831

1 Introduction

The Internet of Things (IoT) is now widespread in contemporary life, enabling continuously broadening "smart" functionality via a multitude of inter-connected and/or internet-connected devices. They range from simple personal and household gadgets [70] to industrial and agricultural machinery [24]. This proliferation of interconnected devices offers considerable benefits, including higher efficiency and better user experience. Identical devices are often grouped together in a dedicated network, interacting either directly among themselves, or being coordinated by a central (often cloud-based) server.

Growing popularity of IoT networks presents a significant challenge: How to attain reliable, secure, and efficient communication among (possibly numerous) IoT devices in a network? In many settings, direct device-to-device communication is required [12] and relying on a trusted server is not an option. For instance, a collection of smart-home devices can collectively monitor the environment and autonomously take actions, e.g., a temperature sensor can directly communicates with a thermostat to adjust room temperature based on real-time data, without any intermediaries. Such direct interaction is crucial for efficient operation of IoT devices.
At the same time, IoT devices have become attractive targets for attacks, especially via remote software compromise. The infamous Mirai botnet [4] that in 2016-1018 infected and zombified (for DoS purposes) a great number of consumer routers and IP cameras is one prominent example of such attacks. The threat is exacerbated by the immense popularity of just a few device types, e.g., Amazon Echo, Ring doorbell, Google Home, Blink cameras, etc.

Problem Statement. How to establish secure device-to-device communication in IoT networks [75] in the presence of possible device compromise?
We assume that, once a device is compromised, the adversary can fully control it by arbitrarily altering its functionality. Without appropriate security measures, a compromised device can negatively impact other devices, e.g., by reporting false measurements that

lead to inappropriate actions. Also, the adversary can introduce rogue devices, which can report false data to other devices and corrupt the entire network.

One straightforward solution is to adopt well-known security techniques, such as public key cryptography. However, this is too computationally intensive (in terms of time and resource usage, e.g., RAM size) for resource-constrained IoT devices. Further, secure storage of secrets (e.g., RNG seeds and private and/or shared keys), requires complex specialized hardware that is not generally unavailable on lower-end IoT devices, such as trusted execution environments. Moreover, even without these two issues, most public key cryptographic techniques are not *future-proof*. Quantum computers [49], using algorithms such as Shor's [69], can efficiently solve problems that form the basis of security for many public key techniques, e.g., integer factorization and discrete logarithms. Specifically, popular public key methods, such as those based on elliptic curves [1] or RSA [61, 71] become vulnerable in a post-quantum world. Therefore, there is a critical need for alternative security techniques that enable future-proof secure device-to-device communication in IoT networks.

PUF-based Authentication. IoT devices can implement Physically Unclonable Functions (PUFs) to derive a unique hardware fingerprint for each device. Each PUF's uniqueness arises from natural variations in the hardware manufacturing process. It is invoked by a challenge to which a PUF responds with a unique identifier specific to its hardware and that challenge. Coupled with the challenge, the unique response forms a challenge-response pair (CRP). By varying the challenge, a given PUF can produce multiple CRPs, which act as device fingerprints and can be used to address the inter-device trust in IoT networks. When two devices interact, authenticity of communicated information can be validated using their respective PUFs. Overall, PUFs enable IoT devices to generate confidential information, such as keys, on demand, which can then be used for authentication. Thus, a PUFs can support authentication without any secure storage on a device.

Existing Methods. There are numerous PUF-based authentication methods [2, 3, 7–10, 15, 17, 19, 21, 23, 35, 37, 39, 44, 48, 56, 72, 73, 77, 81, 82]. Some of them depend on a trusted server to maintain valid CRPs of each PUF. This dependency increases computational and maintenance overhead for the IoT network and creates a single point of failure as well as an attractive attack target. Other techniques that do not require a trusted server [11, 18, 22, 25, 41, 45] are computationally intensive, resource-heavy, or offer weaker security assurances. Since IoT devices tend to have limited hardware resources, such techniques are impractical for many real-world IoT applications. Furthermore, these many prior techniques use cryptographic algorithms susceptible to quantum computing, thus making them unsuitable for future-proof security.

Desired System. An ideal solution should allow secure communication between mutually suspicious IoT devices, without relying on a trusted server. Hence, the solution must support direct device-to-device authentication while (1) avoiding excessive computational load, (2) not requiring trusted hardware, and (3) using quantum computing-resistant cryptography.

Contributions. This paper makes the following contributions:

- We propose AuthentiSafe, a novel and lightweight PUF-based device-to-device authentication scheme tailored for resource-constrained IoT devices. AuthentiSafe eliminates the reliance on a trusted server storing confidential data. Furthermore, AuthentiSafe is the first approach considering the limited computational power and storage capabilities of IoT devices, making it a practical solution for secure authentication in real-world setups. Additionally, the approach stands in it's future-proof nature as it solely relies on methods robust against post-quantum attackers.
- In AuthentiSafe, IoT devices generate confidential PUF responses which are converted into publicly verifiable information that can be used for message authentication using one-time signatures.This way AuthentiSafe obviates the need for a trusted server.
- AuthentiSafe uses one-time signatures which rely on lightweight cryptographic hash functions. This results in efficient processing of PUF responses and message authentication. The exclusive reliance on one-way hash functions makes AuthentiSafe post-quantum secure.
- AuthentiSafe takes advantage of cryptographic accumulators (Merkle trees [47]) to compress one-time signature data into fixed-size representations for creating public identifiers for IoT devices. Use of Merkle trees as accumulators allows AuthentiSafe to efficiently represent the public identifiers of one-time signatures while facilitating sub-linear verification, suitable for IoT devices with meager resources.
- We conducted a systematic evaluation of AuthentiSafe on two off-the-shelf IoT development boards. AuthentiSafe exhibited significant performance advantages, achieving authentication and verification times of 13ms and 33ms, respectively, when using cryptographic hardware acceleration. This represents a runtime improvement of 11 to 17 times in comparison with competing methods. Without hardware accelerators, the measured times were 177ms and 230ms, corresponding to a 6 to 9-fold improvement. These results confirm AuthentiSafe's feasibility and efficiency in real-world IoT scenarios.

Overall, we propose the first scheme offering device-to-device authentication while meeting key requirements: avoiding excessive computations, not relying on trusted hardware, and ensuring future-proof security against post-quantum attacks.

Outline. The remaining part of the paper is organized as follows. Sect. 2 provides background information. Next, Sect. 3 outlines the use case and details the challenges. The proposed framework AuthentiSafe is introduced in Sect. 4, followed by a comprehensive security analysis in Sect. 5. Sect. 6 presents results of the experimental evaluation, followed by the discussion in Sect. 7. Lastly, Sect. 8 overviews related work, and Sect. 9 concludes the paper.

2 Background

Below, we provide the necessary background information about AuthentiSafe's building blocks. First, we discuss PUFs and their role in authentication. Then, we introduce One-Time Signatures in general in and the Winternitz scheme as instantiation. Lastly, we depict cryptographic accumulators with focus on Merkle trees.

Physical Unclonable Functions (PUFs). A PUF is a hardware security primitive that exploits the inherent variabilities due to manufacturing imperfections [34]. These distinct variations can be measured and used to create unique outputs, so-called responses, when provided with inputs, called challenges, thus forming challenge-response pairs (CRPs). It has been observed that multiple PUFs of the same type produce vastly different responses when queried with the same challenge. This phenomenon is attributed to the fact that the response depends on both the specific challenge and on the unique physical properties of a given PUF. It can thus be assumed that the probability of two PUFs of the same type generating the same response to a given input challenge is extremely low. Consequently, PUFs have been widely employed in a variety of security applications, including device authentication, key generation, and intellectual property protection [42, 54, 83]. Unlike traditional cryptographic keys, PUFs are not stored in memory and hence cannot be copied or tampered with [67]. PUFs are classified as weak or strong, based on the number of unique CRPs[46, 62] they can generate. Weak PUFs can generate a limited number of CRPs, usually linearly with respect to the challenge size. In contrast, strong PUFs can generate an exponential number of potential CRPs [20].

PUF-based Authentication. PUF-based authentication is an interactive protocol between a prover and a verifier, designed to confirm the former's possession of a specific PUF. The verifier challenges the prover to generate the correct response for a challenge, thereby authenticating the PUF. PUF-based authentication schemes have two primary phases:

Enrollment Phase. The verifier generates a set of random (and distinct) challenges and requests the corresponding responses from each prover's PUF. The resulting CRPs are securely stored in a database maintained by the verifier. This one-time process must be conducted in a secure environment to ensure confidentiality and integrity of the responses. It is typically done by the device manufacturer prior to device deployment.

Authentication Phase. For each authentication attempt, the verifier selects a random CRP from the database and transmits the challenge to the prover. The prover invokes its PUF with this challenge, obtains a response, and sends it to the verifier. Authentication succeeds if the received response matches that in the CRP. To prevent replay attacks, the current CRP is removed from the database after each authentication attempt.

One-Time Signatures. One-Time Signatures (OTS) offer a unique approach to ensure authenticity and integrity. OTS enable secure message signing using a key-pair: A public key for verification and a private key for signing. The term "one-time" stems from the constraint that each key-pair is used to sign only a single message. OTS schemes are usually based on a cryptographic one-way hash function. This ensures that anyone can verify a signed message using the public key (hash of a private key), while only the private key holder can generate a valid signature. The main benefits of OTS schemes are simplicity and efficiency. They typically require much less computation than traditional public key-based signature schemes, such as RSA [60] or (EC)DSA [53]. They thus represent an attractive choice for devices with limited computational resources. However, an important limitation of OTS schemes is the need to

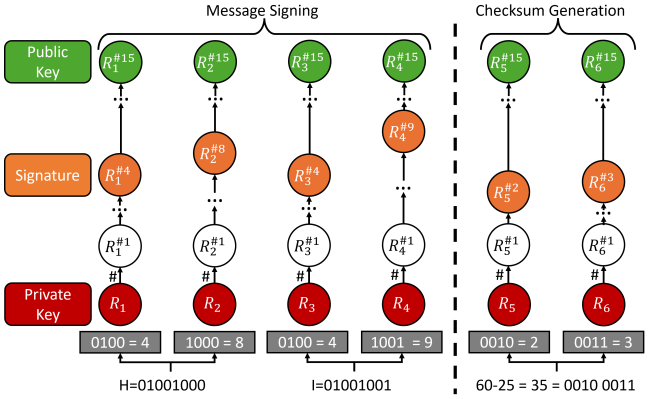

Figure 1: Exemplary overview of the Winternitz OTS Scheme.

generate a new key-pair for every message. This requirement poses challenges in term of key management and storage. Due to this constraint, OTS schemes are less flexible than traditional public key schemes mentioned above. In this work, we integrate OTS with PUFs to develop a novel authentication system. PUFs can produce numerous unique key-pairs as CRPs, thus addressing the OTS limitation. Furthermore, using OTS allows us to transform confidential CRP databases into publicly verifiable data. Our system is instantiated using the Winternitz OTS scheme described below.

Winternitz OTS (WOTS). The WOTS scheme [16] is a hash-based OTS designed to provide secure and efficient signatures. It relies solely on cryptographic hash functions, making it resistant to quantum attacks. This makes WOTS a valuable primitive in post-quantum cryptography. Also, because each key-pair is only used once, WOTS avoids vulnerabilities associated with traditional signature schemes where keys are reused. The architecture of the WOTS scheme is illustrated in Fig. 1. Overall, the scheme consists of three main phases:

Key Generation. Here, the signer generates a private key, consisting of a set of random numbers (red in Fig. 1). For each element in this set, a corresponding public key element is derived by iteratively applying a cryptographic hash function a predetermined number of times, e.g., 15 times in Fig. 1.[1] This process results in individual hash chains originating from each private key element. The collective set of hashed values forms the public key (green in Fig. 1).

Signing. The signing process begins by hashing the message to produce a message digest, which consist of the two characters, H and I at the bottom in Fig. 1. This digest is then converted to bits, in Fig. 1 with ASCII. The bit sequences are then divided into multiple parts according to the Winternitz parameter w, which dictates the number of bits processed simultaneously. In Fig. 1, we use $w = 4$, resulting in the digest being split into equal 4-bit parts. Each part determines the number of times the hash function is applied to the corresponding private key element, e.g., four times for the first 4-bit sequence, as illustrated by the grey square boxes at the bottom of Fig. 1. Subsequently, the hash function is iteratively

[1]We hash 15 times because the example uses 4-bit messages parts. Those can encode the numbers from 0 to 15, hence we hash $2^4 - 1 = 15$ times.

applied the determined number of times, forming the signature (orange in Fig. 1). A crucial component of the WOTS scheme is the checksum (depicted on the right in Fig. 1). The checksum is used to verify the integrity of the signed message. It is computed by summing the values derived from the message digest parts (in the gray square boxes) and subtracting them from a predetermined maximum value, which is the amount of hashes required to generate the public key (green in Fig. 1), which is $4 \cdot 15 = 60$ in Fig. 1. This checksum is included as part of the (orange) signature. In Fig. 1, $4 + 8 + 4 + 9 = 25$ hash operations have already been conducted to generate the signature, which is the sum of all hashes used in its construction. Since a total of 60 hashes were used to form the public key (green), 35 hash operations remain. To prevent message manipulation, this number is appended to the message digest and processed similarly to form the checksum.

Verification. The verifier performs the remaining hash functions needed to generate the public key elements from the signature (orange to green in Fig. 1). The verifier also recomputes the checksum from the received message parts and compares it to the checksum included in the signature, ensuring that no part of the message has been altered. If the checksum is correct and the re-generated public key matches the public key initially provided by the signer, the signature is valid.

The Winternitz parameter defines how to interprete the input message, e.g., one input can have four bits or eight bits. Accordingly, the length and the number of the hash chains are adapted. Essentially, the Winternitz parameter directly influences both the signature length and the computational complexity of signature generation and verification. A higher Winternitz parameter value reduces the signature length, which minimizes bandwidth. However, this incurs increased computational cost for both signature generation and verification. Conversely, a lower Winternitz parameter value results in longer signatures but speeds up signing and verification. This trade-off is advantageous in scenarios where computational resources are limited (e.g., IoT devices) and bandwidth is less critical. In practice, Winternitz parameters of four or eight are commonly used, since they offer a good balance between computational effort and signature size taking into account factors such as available computational resources, bandwidth, and storage capacity.

Cryptographic Accumulator. Cryptographic accumulators are compact data structures that use cryptographic primitives to achieve sub-linear time complexity for set membership operations, e.g., efficiently verify whether a given element is part of a set. Initially introduced by Benaloh and De Mare [13] for document time-stamping, cryptographic accumulators have since been adopted in various application domains. Their use can significantly reduce the overhead of set membership checking while minimizing storage requirements. A prominent example is the Merkle tree [47], a (usually binary) tree where leaves represent hashes of set elements. The value of each non-leaf node is computed in a bottom-up fashion by hashing the values of its children. Thus, the final value, called a root, aggregates all other nodes in the Merkle tree. To prove that a given element (leaf) is part of the set represented by an Merkle tree, it is necessary to convince the verifier that leaf, plus values of all sibling nodes on the path to the root form that root. This proof is logarithmic in terms of the set size, i.e., the number of leaves. This is in strong contrast with proving membership by sending a signed list of all elements or hashes thereof.

3 Problem Statement

We consider large IoT networks that connects heterogeneous IoT devices from mutually mistrusting stakeholders. These stakeholders play crucial roles in deploying, operating, and maintaining their IoT devices within a heterogeneous and extensive ecosystem. The IoT devices within the network interact both within and across their stakeholders' boundaries to generate, collect, and share information directly. App. Fig. 5 shows an example of such a network.

Problem. In complex IoT networks, trust is a critical issue. Without safeguards, the potential for malicious activities, such as message manipulation, increases significantly and can compromise the overall reliability and security of the IoT network. Mistrust among stakeholders complicates establishing trust, influenced by several correlated challenges discussed below.

C1 - Distributed Nature of IoT. The distributed nature of IoT networks complicates consistent security management across the entire system, requiring decentralized approaches to authentication and access control. Hence, each stakeholder typically operates under distinct administrative controls and adheres to varying security policies, making it challenging to establish a uniform trust landscape across the network. This results in the absence of universally accepted security standards and protocols, which hinders interoperability and comprehensive security implementation across devices and platforms, particularly those from different vendors. The lack of standardization affects security and impacts overall efficiency and effectiveness of IoT ecosystems.

C2 - Central Trusted Party. However, trust can be facilitated through the implementation of centralized trusted parties provided by individual stakeholders. These trusted entities serve to verify device authenticity and establish trust across a decentralized network. In networks operated by mutually distrustful parties, the establishment of consensus on a shared entity to manage confidential information presents a significant challenge due to conflicting interests and security concerns of different stakeholders. Hence, stakeholders often implement their own solutions tailored to their specific device ecosystem. While these solutions may be effective within a limited scope, they are inherently unable to extend to a large and heterogeneous IoT network that interconnects various heterogeneous devices from multiple vendors and stakeholders.

C3 - Network Fluctuation. The trust challenge is further compounded by the dynamic nature of IoT networks, where devices frequently join or leave, and network topology evolves continuously. This fluidity demands trust establishment mechanisms that are robust and adaptable to changing networks.

C4 - Constrained Resources. The diversity of device types and capabilities complicates the implementation of uniform and robust security measures. Resource-constrained IoT devices limit the complexity of security protocols, requiring adaptive solutions that scale across various devices while maintaining a consistent level of protection.

C5 - Adversarial Hardware Access. Furthermore, IoT devices often operate in unattended and/or remote locations, which increases their susceptibility to tampering and physical attacks. This prompts

additional security concerns that traditional network security measures do not adequately address.

C6 - Post-Quantum Threat. Quantum computing threatens current cryptographic systems, adding urgency to develop quantum-resistant, efficient trust mechanisms for IoT networks.

Desired System. Our goal is to establish distributed trust, enabling direct interactions between mutually mistrusting devices within an IoT network. The proposed scheme must address the challenges mentioned above. Thereby, post-quantum security is crucial, ensuring long-term protection of communications and data integrity against quantum-capable adversaries. Additionally, the solution must support heterogeneous hardware, operating efficiently across various devices without extensive customization. It should also function without relying on advanced secure hardware, such as trusted execution environments or secure storage, to accommodate the resource limitations of many IoT devices.

Threat Model. Our adversary model combines the Dolev-Yao framework [27] with the non-volatile memory adversary paradigm [6]. According to the former, the adversary is assumed to control communication channels and is, thus, capable of replay, interception, and spoofing of messages. Based on the latter, the adversary is also assumed to have physical access to devices, which enables extraction of any information stored in non-volatile memory. This model is particularly pertinent to IoT environments, as IoT devices are frequently deployed in public or semi-public spaces, which increases their vulnerability to physical tampering and unauthorized access. Further, IoT devices are often designed with simplicity and cost-effectiveness in mind. Thus, they frequently lack secure or tamper-resistant hardware, which makes them more susceptible to direct physical access or manipulation. By considering an adversary that controls the network and is capable physical memory extraction, our model reflects realistic and significant threats faced by IoT devices in real-world deployments.

Our approach uses PUFs and cryptographic hash functions. Security of each of these underlying techniques is assumed and considered outside the scope of this work. Additionally, relay attacks, that can be mitigated by distance bounding techniques are excluded, which is a common assumption for authentication schemes. Similarly, Denial of Service (DoS) attacks are excluded. We note the goal of authentication protocols is to verify the identities of users or entities. They are not intended to protect against or mitigate, DoS attacks, which target resource availability. Effective mitigation of DoS attacks requires specialized countermeasures tailored to specific attack characteristics, distinct from the mechanisms used in authentication protocols. Lastly, although side-channel attacks would have a limited impact on our protocol, they remain outside the scope of this discussion. We assume that the manufacturer produces honest, non-malicious hardware and adheres to the enrollment protocol.

4 Design

The primary objective of AuthentiSafe is to provide a lightweight, post-quantum secure authentication mechanism for resource constrained IoT devices within a large, unified network of interconnected devices from multiple mutually mistrusting stakeholders. AuthentiSafe is specifically designed to operate without a central trusted party and is sufficiently lightweight to be deployed across a diverse and extensive array of heterogeneous devices. Our scheme's efficiency enables both prover and verifier capabilities even on highly resource-constrained IoT devices, effectively addressing the challanges outlined in Sect. 3. Therefore, AuthentiSafe integrates Physical Unclonable Functions (PUFs), one-time signature (OTS) schemes, and cryptographic accumulators, as introduced in Sect. 2.

4.1 Overview

In AuthentiSafe, IoT devices leverage their internal PUF to generate confidential information in the form of challenge-response pairs (CRPs) used for authentication purposes. Those CRPs can be considered similar to a cryptographic key, but instead of being stored on the device, they are generated on demand. Our scheme uses a freshly generated CRP as basis for each authentication process. Hence, even if a PUF response is compromised at a certain time, it is only valid for a single authentication process. To allow verification of the authentication without relying on the impractical overhead of maintaining large confidential CRP databases, AuthentiSafe effectively transforms previously confidential CRP information into public information using one-time signature (OTS) schemes. This public data does not leak any information about the underlying confidential data, namely the concrete CRPs, but enables the verifier to authenticate. Thereby, the OTS solely relies on cryptographic hash functions, which are post-quantum secure. Additionally, we use a cryptographic accumulator to organize this public information, reducing its size and improving the verification performance for the verifier. Accumulators offer a space-efficient representation of data as well as sub-linear verification effort, making them particularly well-suited for resource-constrained IoT devices.

Overall, AuthentiSafe offers a flexible solution suitable for resource-limited IoT devices, combining on-demand PUF-based authentication with efficient public verification mechanisms. Therefore, it offers efficient PUF-based authentication without relying on large and confidential CRP databases. PUF-based authentication schemes typically consist of two phases: an enrollment phase and an authentication phase (cf. Sect. 2). Our scheme follows this established structure, with both phases designed to leverage the unique properties of PUFs while addressing the specific needs of IoT devices.

Enrollment Phase. AuthentiSafe conducts the standard enrollment phase, generating CRPs from its PUF within a secure environment during the manufacturing process. However, our scheme goes beyond the traditional enrollment approach, which typically stops at storing the generated CRPs confidentially in a secure database hosted by the verifier. Instead, AuthentiSafe takes the generated raw and confidential PUF responses and transforms them into information that is still related to the original PUF response and allows verification, but does not leak information about the underlying PUF response. To achieve this, we utilize a OTS scheme, transforming each PUF response into the public key of a OTS. This transformed information is then stored in a publicly accessible repository, ensuring that the authentication process does not rely on confidential data stored on a trusted server. Consequently, the manufacturer's role is confined to the initial setup: generating the CRPs, converting them into individual OTS public keys, and publishing them. The

manufacturer has no involvement in subsequent authentication processes. Upon completing enrollment, all stored PUF responses are securely deleted by the manufacturer. This approach establishes a foundation for a more transparent and decentralized authentication system. Additionally, to ensure efficient verification, we accumulate the individual public identifiers into a cryptographic accumulator to allow for efficient sublinear verification. This structure not only enhances the scheme's efficiency but also provides a compact representation of the device's authentication data.

Authentication Phase. This phase of AuthentiSafe involves two parties: the prover and the verifier, mirroring standard PUF-based authentication processes. The protocol's primary objective is to enable the prover to demonstrate its identity to the verifier by proving possession of the actual PUF. This phase is designed to leverage the unique properties of PUFs while ensuring security and efficiency. The prover holds a counter value (described later in Sect. 4.2), which serves as PUF challenge for the current authentication process and is subsequently incremented. To pass the authentication process in AuthentiSafe, the prover must produce a valid OTS of the provided authentication challenge using its PUF. The process unfolds as the prover first generates the confidential PUF response. This response is then used for the creation of the private key in a OTS scheme (red in Fig. 1). Subsequently, the message signature (orange in Fig. 1) is generated. The message and the constructed signature is subsequently transmitted to the verifier. Upon receiving the signed message, the verifier first creates the public key of the OTS scheme (green in Fig. 1). Following this, the verifier checks if the public key of the used OTS actually belongs to the prover. This is done by fetching the public cryptographic accumulator created during the enrollment phase and conducting a membership set test. If the test is passed, the authentication is considered successful. This approach offers several advantages. It eliminates the need for storing secret information on the verifier's side, significantly enhancing security. The use of OTS ensures that each authentication attempt is unique, effectively preventing replay attacks. The cryptographic accumulator structure allows for efficient verification and compact representation of the enrolled elements, which is particularly beneficial for resource-constrained devices. Moreover, the public nature of the verification data enables a more transparent and potentially decentralized authentication process, aligning with modern trends in secure system design. By combining PUF-based key generation with OTS and cryptographic accumulators for verification, AuthentiSafe provides a robust, efficient, and secure authentication mechanism. This makes it particularly suitable for IoT devices and other resource-constrained environments where traditional cryptographic methods might be impractical or too resource-intensive.

4.2 Counter-Based PUF

Our PUF implementation incorporates an important feature: an integrated monolithic counter that enhances the security and reliability of the authentication system. This counter, matching the challenge length (e.g., 128 bits), is embedded directly into the PUF hardware and automatically increments with each query to the PUF. The current value of the counter serves as the challenge input to the PUF, ensuring that no challenge is never used twice. This aligns with PUF constructions previously proposes in works such as [57] or [29]. This mechanism ensures that a new PUF response is used for every authentication process. While the counter progression is deterministic, the PUF responses remain unpredictable due to the inherent properties of the PUF, providing a perfect balance of reliability and security. This approach eliminates the need for external challenge generation or storage, simplifies the overall system design, and maintains the unpredictability and unclonability properties of the PUF while providing a structured authentication sequence. The counter-based challenge system aligns well with our implementation of OTS and cryptographic accumulator verification, as it naturally produces a sequence of unique challenges that can be efficiently managed and verified within our authentication framework. It is important to note, that the final step in the enrollment phase involves a critical security measure concerning the counter. The manufacturer resets the PUF counter to its initial state and then permanently disables the reset mechanism by blowing a fuse. This irreversible process is of utmost importance as it ensures the integrity of the counter and prevents any unauthorized manipulation of its initial state. By rendering the reset mechanism inoperable, we significantly enhance the overall security of the authentication system, making the device ready for deployment in real-world scenarios.

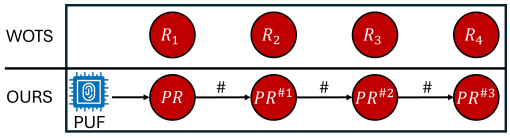

Figure 2: WOTS private keys are created by hashing PUF responses instead of using random values.

4.3 Details

In the following sections, we will delve into the detailed mechanisms of AuthentiSafe. We will begin by precisely outlining the enrollment and authentication processes, followed by a discussion on the publicly accessible data.

Enrollment Phase. AuthentiSafe's enrollment process involves key steps to establish a secure foundation for future authentication. This phase is essential for setting up the infrastructure needed for secure device communications. Initially, the counter of the PUF is set to 0, establishing a clean starting point. The enrollment process then proceeds by querying the PUF a predetermined amount of times. This sequential querying generates a series of PUF responses (PRs). Each of these PRs is unique due to the inherent properties of the PUF and the incrementing counter value used as a challenge. Following the generation of the PRs, each PR is utilized to create a public key for a OTS. For AuthentiSafe, we chose the Winternitz One-Time Signature (WOTS) scheme (cf. Sect. 2), a variant of OTS renowned for its efficiency, strong security properties, and post-quantum resistance. These characteristics make it particularly well-suited for our PUF-based authentication system designed for resource-constrained devices. Instead of using random values as private key in WOTS (red in Fig. 1), we use the PRs to generate a unique private key. Thereby, as visualized in Fig 2, the PR is attractively hashed with a hash function to construct random values

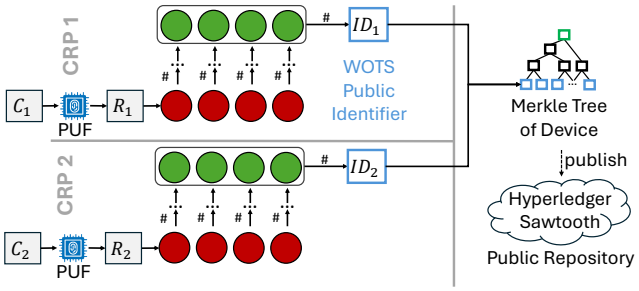

Figure 3: Construction of public data by AuthentiSafe.

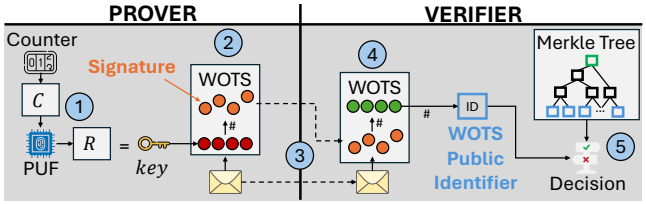

Figure 4: Authentication phase of AuthentiSafe.

that are based on the PR. While the implementation of WOTS and AuthentiSafe is independent from a concrete hash function, this hash function must be different to the one used in WOTS, but stem from the same function family.[2] To derive a WOTS public key, we follow the regular WOTS scheme (including checksum generation). This method ensures that a single PUF response is securely extended and processed to form a secure and reliable WOTS private and subsequently public key (red and green in Fig. 1). By carefully choosing the Winternitz parameter, we can balance the trade-off between signature size and computational complexity, adapting to the specific requirements of resource-constrained IoT devices. This approach not only provides strong security guarantees but also efficiently uses the limited resources of IoT devices.

The enrollment phase results in a series of WOTS public keys (green dots in Fig. 3). To enhance efficiency and reduce storage requirements, the public key of each WOTS are aggregated into a single, fixed-size representation by hashing the key values with the WOTS hash function. This hash results in the WOTS public identifier (blue boxes in Fig. 3). The identifiers of all CRPs of one device are then accumulated into a cryptographic accumulator, which significantly reducing the amount of data that needs to be stored or transmitted for verification purposes. While various accumulator structures exist and could be used, we instantiated AuthentiSafe using a Merkle tree [47] since it perfectly aligns with the nature of WOTS; both schemes are fully based on cryptographic hash functions. In line with modern decentralized security practices, the accumulator value, the Merkle root (green box in Fig. 3), is then published on a distributed ledger. This approach ensures transparency and makes the verification data publicly accessible while maintaining its integrity. Concurrently, the individual public identifiers are stored using the InterPlanetary File System (IPFS) [26].[3] With all necessary information now publicly available, the manufacturer no longer needs to retain PUF responses and has deleted all data related to the enrollment process. This enrollment process establishes a secure, efficient, and tamper-resistant foundation for our PUF-based authentication system. It leverages the unique properties of PUFs, the security of WOTS, the efficiency of hash functions, the space-efficiency of cryptographic accumulators and the robustness of distributed storage systems to create a novel and lightweight solution for secure device authentication. This marks the conclusion of the enrollment phase, transitioning the device to the authentication phase.

Authentication Phase. The authentication process involves a sequence of carefully coordinated steps to ensure secure communication and verification between the prover and the verifier. The process, as visualized in Fig. 4 unfolds as follows: 1) To initialize authentication, the prover reads the current value of the integrated counter, which serves as the input to the PUF. The PUF generates a corresponding response based on this counter value. Importantly, querying the PUF automatically increments the counter, ensuring a unique challenge for each authentication attempt. The generated PUF response is used to generate the private key of the WOTS scheme (red dots in Fig. 4). Thereby, the generation of the private key follows the same mechanism used in the enrollment phase. 2) Next, the WOTS private key is used to generate the message signature within WOTS (orange dots in Fig. 4). 3) Once finished, the message together with the signature is transmitted to the verifier. 4) Upon receiving the signed message, the verifier first computes the public keys of the WOTS (green dots in Fig. 4). This allows the verifier to check the checksum of the message essentially detecting any potential tampering. If the checksum is valid, the verifier creates the WOTS public identifier (blue box in Fig. 4) by hashing the public key. 5) The verifier must confirm that the calculated WOTS public identifier has been previously enrolled. Therefore, the verifier accesses the cryptographic accumulator of the prover's device, which contains all public identifiers registered during enrollment. A successful membership set test confirms the prover's identity and message authenticity. To do so, the verifier requests the corresponding Merkle path for the generated identifier value and uses it to generate the Merkle root by hashing the received elements starting with the identity value along the path. If the generated Merkle root matches against the prover's Merkle root generated during the enrollment phase, the verifier can be sure that the authentication is valid. Notably, IoT devices, initially serving as verifiers, can also be enrolled during the manufacturing phase, enabling them to function as provers when needed. This dual functionality enhances the flexibility and security of the IoT network.

5 Security Analysis

This section provides an informal security analysis of AuthentiSafe by examining potential attacks, assessing their impact, and discussing possible mitigation strategies. We focus solely on the security of AuthentiSafe itself and exclude any vulnerabilities related

[2]We use a different hash function, as otherwise parts of the private key could be reflected in the signature. Different hash functions from the same family share similar characteristics but produce different outputs for the same input, enhancing security by reducing the risk of collision and pre-image attacks.

[3]IPFS, a distributed file system, provides a robust and decentralized storage solution, further enhancing the system's resilience against potential attacks or data loss. Alternative storage solutions are discussed in Sect. 7.

to the underlying technologies, such as distributed ledgers, PUF implementations, or cryptographic primitives.

Replay Attack. Replay attacks pose a significant threat to authentication schemes, particularly when an adversary gains control over the network. In such an attack, the malicious actor intercepts valid authentication messages and retransmits them later to impersonate a legitimate party. This can be especially effective against protocols that lack measures to ensure message freshness. However, this vulnerability can be effectively mitigated by incorporating a verifier-supplied nonce into the authentication process. For each authentication attempt, the verifier generates a unique, random nonce and sends it to the prover. The prover must include this nonce in the message to be signed. When the verifier receives the signed message, it checks both the validity of the signature and that the included nonce matches the one it just sent. This approach ensures that the received information is fresh and not replayed. Even if an attacker intercepts a valid authentication message, it cannot be successfully reused because it would contain an outdated nonce. The verifier would recognize that the nonce in the replayed message doesn't match the current nonce it issued, thus detecting the replay attack.

Message Integrity. An adversary attempting to compromise the authentication scheme might intercept a message signed by an honest prover and try to alter its content. However, this attack is effectively prevented by the checksum mechanisms built into the signature scheme. If the adversary modifies the message and attempts to adjust the signature by applying additional hash operations to certain parts, it would inevitably disrupt the validity of the signature's checksum. The resulting signature would be invalid and easily detected by the verifier in step four of Fig. 4. To create a valid signature for the altered message, the adversary would need to adjust the checksum accordingly. This adjustment would require reducing the hash values of the checksum components, which is equivalent to reversing a cryptographic hash function. Given the one-way nature of cryptographic hash functions, this operation is computationally infeasible even for quantum computers.

Key Forgery Attack. A key forgery adversary might attempt to undermine the authentication system by generating their own WOTS private and public key and using it to sign messages, impersonating a legitimate prover. Although this approach allows the adversary to create a technically valid signature, the attack is ultimately futile. During the verification process, the system checks the public identifier of the used WOTS against a Merkle tree generated for the honest prover during the enrollment period. Since the adversary's forged key was not part of this initial enrollment, its public identifier is absent in the Merkle tree. This mismatch enables the verifier to detect and reject the adversarial attempt, effectively neutralizing the key forgery attack.

Manipulation of Public Information. The security of the authentication system extends to the protection of public information stored during the enrollment phase. A key component of this protection is the storage of device identifiers, such as accumulators or their root nodes, on a distributed ledger using smart contracts. These contracts are designed to grant exclusive write privileges to the manufacturer, while maintaining public read access. This approach creates a robust safeguard against tampering attempts. While the stored information remains transparent and accessible to all network participants, the restriction on write access effectively prevents unauthorized modifications to the data on the distributed ledger. The system also utilizes IPFS [14] for storing public identifiers of WOTS public identifiers. Although this data could potentially be subject to manipulation, the authentication process incorporates safeguards against such attempts. Any alterations, including modifications to Merkle paths, would result in failed membership tests during verification, leading to the rejection of the authentication attempt (step 5 in Fig. 4). Notably, while such tampering attempts could potentially disrupt service availability, they fall under the category of Denial of Service (DoS) attacks, which are beyond the scope of the threat model outlined in Sect. 3.

Extract Non-volatile Memory. While an adversary with physical access to an IoT device might attempt to extract sensitive information stored in non-volatile memory through techniques like memory dumping, AuthentiSafe is designed to mitigate such risks. The key security feature lies in its on-demand generation of authentication data in the form of PUF responses, rather than persistent storage of sensitive data. Hence, confidential authentication credentials are not stored in the device's non-volatile memory. Instead, they are generated on demand during the authentication process when required, leveraging the unique physical characteristics of the device and are not stored directly on the prover device. Thereby, the adversary cannot gain any confidential information by extracting non-volatile memory. The prover device does maintain an integrated counter, which an adversary could potentially read. However, this counter alone provides no meaningful insight into the authentication process. The actual authentication response is intrinsically tied to the device's physical attributes, making it resistant to simple extraction or replication.

Temporary Device Compromise. In IoT environments, devices may become temporarily compromised due to software vulnerabilities. However, these vulnerabilities can be patched, restoring the device to its secure state. Schemes that generate an asymmetric key from the PUF response, such as those proposed in [25, 31, 32, 68, 76], are vulnerable because the same PUF response is reused for every authentication session. Once an adversary gains access to this PUF response, they can indefinitely impersonate the device, even after it has been cleaned. In contrast, AuthentiSafe significantly limits the adversarial impact by using a new PUF response for each authentication session. Each session employs a distinct element from the PUF to derive a fresh private key. If an adversary gains access to the device, they can only compromise the elements of the currently used WOTS key. However, for the next authentication process, a fresh PUF response is used, rendering the previously compromised information useless. This approach enhances security by ensuring that past compromises do not have a lasting impact, providing robust protection against temporary device compromises.

6 Implementation & Evaluation

Below, we first outline our hardware and software configuration, detailing the microcontrollers used and the relevant works for comparison. Next, we offer a more in-depth discussion of the implementation details in Sect. 6.1. Finally, Sect. 6.2 and Sect. 6.3 present

Table 1: Authentication runtime (in s) and clock cycles (10^6).

LPC55S69-EVK board	Accelaration		No Acceleration	
	Cycles	Time	Cycles	Time
Felicetti *et al.* [31]	65.12	0.43	320.09	2.13
Samra *et al.* [63]	34.59	0.23	162.71	1.09
Petzi *et al.* [51]	31.92	0.21	329.05	2.19
AuthentiSafe ($w = 4$)	1.80	0.01	24.59	0.16
AuthentiSafe ($w = 8$)	12.05	0.08	166.17	1.11

Table 2: Runtime comparison in seconds.

Atmel ATmega1284P board	Authentication	Verification
Felicetti *et al.* [31]	11.68	7.30
Samra *et al.* [63]	6.80	7.30
Petzi *et al.* [51]	6.90	7.20
AuthentiSafe ($w = 4$)	2.59	2.87
AuthentiSafe ($w = 8$)	20.90	21.32

the runtime performance evaluation and the assessment of memory consumption, respectively.

Hardware. To demonstrate AuthentiSafe's efficiency and practicality, we implemented the scheme on two distinct IoT development platforms, representing a range of modern IoT device capabilities. The first platform, the LPC55S69-EVK from NXP Semiconductors, is a 32-bit microcontroller and features an Arm Cortex-M33 processor operating at up to 150 MHz, with 640 KB of flash memory and 320 KB of SRAM. The second platform, the Atmel ATmega1284P-XPLD, represents a highly resource-constrained environment. This 8-bit microcontroller runs at 20 MHz, with 128 KB of flash memory and 16 KB of SRAM, contrasting significantly with the LPC55S69-EVK in terms of computational resources. We implemented a software-simulated PUF to mimic hardware behavior, enabling protocol evaluation with perfect PUF responses, thereby focusing on protocol features. Similar PUF outputs can be achieved with hardware PUFs and appropriate error correction, as outlined in Sect. 7.

Software. To demonstrate the advantages of AuthentiSafe, we implemented three recently proposed PUF-based authentication schemes on both IoT development platforms, comparing their performance with AuthentiSafe. The first scheme, proposed by Felicetti *et al.* [31], uses PUF responses to generate ECDSA signatures [74] for message authentication. This method derives the ECDSA private key directly from a PUF response, linking the device's hardware properties to its cryptographic functions. The second scheme, introduced by Samra *et al.* [63], features an innovative key management approach. It encrypts a private key using AES, with the encryption key derived from the PUF response. This ensures that key decryption is only possible with the correct PUF response, binding the private key's security to the PUF's unique characteristics. The third scheme, proposed by Petzi *et al.* [51], employs zero-knowledge proofs in PUF-based authentication. This method verifies the PUF response's correctness without revealing the actual output, enhancing security by maintaining the confidentiality of the PUF data while enabling robust authentication.

Hardware Accellerator. The LPC55S69-EVK board includes a dedicated cryptographic hardware accelerator, enhancing security and processing efficiency for cryptographic tasks. Specifically, the board features a CASPER hardware accelerator for elliptic curve cryptography and a HASHCRYPT accelerator supporting SHA-1 and SHA-2, including SHA-256. While AuthentiSafe utilizes HASHCRYPT for SHA-256 acceleration, related schemes that rely on elliptic curve cryptography leverage CASPER acceleration to ensure optimal runtime and a fair comparison.

By testing AuthentiSafe on diverse platforms and comparing it with three recently proposed PUF-based authentication schemes, we offer a comprehensive evaluation of its performance, resource utilization, and scalability across various IoT device categories. The following sections detail our implementation methodology, present experimental results, and analyze the scheme's efficiency and adaptability in real-world applications, providing valuable insights into AuthentiSafe's relative merits and potential applications across a range of hardware configurations.

6.1 Implementation

To maintain uniformity and eliminate potential performance discrepancies arising from library variations, we carefully selected and utilized optimal cryptographic libraries for each platform (see details below). We reimplemented previous works, as no source code was publicly available, using C for all implementations to ensure efficiency and portability across embedded systems. This choice leverages C's performance and portability across various embedded platforms. We utilized SHA-256 [28] as the hash function across all schemes, given its widespread adoption and reliability. However, any other hash function can be utilized, offering the same level of security depending on its cryptographic properties. For schemes involving elliptic curve operations, we employed the SECP256R1 curve [40], a NIST-standardized curve recognized for its strong security and computational efficiency. By maintaining consistency in our implementations and carefully selecting platform-appropriate libraries, we aimed to create a level playing field for comparison. This strategy ensures that any observed performance differences can be attributed to the inherent characteristics of each authentication scheme and the capabilities of the respective hardware platforms, rather than discrepancies in library efficiency or implementation techniques.

Used Libraries. For the implementations on the LPC55S69-EVK, we employed mbedTLS [5] as our cryptographic library, while for the Atmel ATmega1284P, we opted for AVR-Crypto-Lib. MbedTLS, developed and maintained by Arm, is a comprehensive and widely respected cryptographic toolkit specifically optimized for embedded devices. Its robust feature set and efficient implementation make it an ideal choice for our microcontroller platform. The AVR-Crypto-Lib is specifically designed and optimized for AVR devices, ensuring that our implementation on this more resource-constrained platform benefits from tailored cryptographic routines. AVR-Crypto-Lib's lightweight nature and AVR-specific optimizations allow for efficient utilization of the ATmega1284P's limited resources.

Table 3: Verification runtime (s) and clock cycles (10^6).

LPC55S69-EVK board	Accelaration		No Acceleration	
	Cycles	Time	Cycles	Time
Felicetti *et al.* [31]	36.55	0.24	332.68	2.22
Samra *et al.* [63]	36.55	0.24	332.68	2.22
Petzi *et al.* [51]	42.90	0.29	331.01	2.21
AuthentiSafe ($w = 4$)	1.58	0.01	21.34	0.14
AuthentiSafe ($w = 8$)	12.16	0.08	187.59	1.25

6.2 Runtime Evaluation

Below, we evaluate AuthentiSafe's runtime against the three other schemes. Notably, these schemes offer lower security since they are not post-quantum secure, while AuthentiSafe relies solely on cryptographic hash functions for post-quantum security. Hence, as the results demonstrate, AuthentiSafe is both more secure and superior in performance.

Autentication Process. The results of our runtime evaluation of the authentication for all four analyzed schemes executed on the LPC55S69-EVK are presented in Tab. 1. The reported execution times are the average results of 100 runs. Given that this board is equipped with a cryptographic hardware accelerator, our evaluation considers both scenarios: with the accelerator active and disabled. For AuthentiSafe, we utilized two different Winternitz parameters, $w = 4$ and $w = 8$, to assess the impact of this parameter. For the other schemes, the same parameters as in the original papers were used. Our analysis clearly demonstrates that AuthentiSafe significantly outperforms all previously evaluated schemes in both scenarios, whether the hardware accelerator is enabled or not. For instance, for $w = 4$, AuthentiSafe needs 0.01s, while others take between 0.21s and 0.43s, equaling an improvement between 95.45% to 97.73%. One interesting observation is that for $w = 4$, AuthentiSafe without hardware acceleration is still faster than all other analyzed schemes with hardware acceleration showing its superiority. Although the execution times increase for $w = 8$, AuthentiSafe continues to outperform most other schemes with hardware acceleration beside Samra *et al.* [63], which has slightly better runtime. For the Atmel ATmega1284P, the runtime evaluation results of the authentication process in seconds are presented in Tab. 2. Since this device lacks cryptographic hardware acceleration, we report only the execution times of the software-only implementation. The results exhibit a pattern similar to those observed on the LPC55S69-EVK. Specifically, AuthentiSafe with a Winternitz parameter of $w = 4$ significantly outperforms all its competitors, achieving execution times that are less than half of theirs. Especially, AuthentiSafe with $w = 4$ ran 2.59s, while others needed between 6.80s and 11.68s, equalling an improvement of 61.91% to 77.82%. This clearly demonstrates the superior efficiency of AuthentiSafe in terms of computational effort. However, increasing the Winternitz parameter to $w = 8$ significantly raises execution time to 20.90s, making it less practical for low-end devices like the ATmega1284P. This suggests that $w = 4$ is a more suitable parameter for such constrained hardware, offering a better balance of efficiency and performance. This evaluation highlights the importance of parameter selection in cryptographic implementations, particularly on resource-limited devices. The choice of $w = 4$ for AuthentiSafe provides a compelling case for its adoption in low-power and low-cost embedded systems.

Verification Process. To evaluate the performance of AuthentiSafe, we used a Merkle path length of 25, corresponding to a Merkle tree size of 2^{25}, which represents a total of 33,554,432 enrolled responses. The results for the LPC55S69-EVK are illustrated in Tab. 3. The evaluation reveals a pattern similar to that observed in the authentication process. AuthentiSafe, with its Winternitz parameter configurations, significantly outperforms the three previously analyzed schemes in both scenarios, with and without hardware acceleration. For instance without hardware acceleration, AuthentiSafe needs 0.14s for $w = 4$, while others take around 2.22s, resulting in an improvement of up to 94.1%. Notably, although the authentication performance of the approach by Samra *et al.* [63] without hardware acceleration was comparable to AuthentiSafe with $w = 8$, as shown in Tab. 1, AuthentiSafe demonstrates significantly superior performance in the verification phase, as reported in Tab. 3, and achieves even greater improvements when hardware acceleration is available. An important observation from our evaluation is that the verification execution times for the schemes proposed by Felicetti *et al.* [31] and Samra *et al.* [63] are identical. This is due to the fact that both approaches utilize the same verification mechanism. These results underscore the efficiency of AuthentiSafe in the verification process, further undermining its advantage over existing schemes, particularly in resource-constrained environments. The results for evaluating the verification process on the ATmega1284P are displayed in Tab. 2. We observed similar results as those obtained during the authentication evaluation. Specifically, AuthentiSafe with a Winternitz parameter of $w = 4$ significantly outperformed all other schemes, achieving execution times that were more than twice as fast as its competitors. This highlights the efficiency and speed advantage of AuthentiSafe in the verification process, making it highly suitable for resource-constrained devices. However, similar to the authentication evaluation, increasing the Winternitz parameter to $w = 8$ resulted in a significantly increased execution time. This emphasizes that while $w = 8$ may offer other benefits such as shorter signature sizes, it is not the optimal choice for low-end devices. Thus, $w = 4$ remains the preferred parameter for instantiating AuthentiSafe on such devices, balancing performance and efficiency effectively. This reinforces AuthentiSafe's advantages, particularly with $w = 4$, in achieving superior verification performance on low-power embedded systems.

6.3 Memory Consumption

This section compares the memory consumption of AuthentiSafe against three other authentication schemes [31, 51, 63] for IoT devices. Given the resource constraints of IoT environments, minimizing memory usage is critical for practical deployment. We assess each scheme's memory requirements during key operations, specifically authentication and verification. The results, based on evaluations conducted on the LPC55S69-EVK, are presented in App. Tab. 4. For AuthentiSafe, two scenarios were analyzed ($w = 4$ and $w = 8$) to examine the impact of this parameter on memory usage.

Tab. 4 shows that in our implementation, all data was allocated on the stack, resulting in no heap usage due to the chosen library. In

contrast, elliptic curve parameters were allocated on the heap because the library generates them. Overall, AuthentiSafe with $w = 4$ required 4030 bytes, improving memory consumption by 9.56% to 50.36% compared to related works, which used between 4456 and 8120 bytes. While $w = 4$ was optimal for runtime, $w = 8$ yielded better memory efficiency, requiring only 2418 bytes. This demonstrates the trade-off between runtime and memory consumption, as AuthentiSafe hashes more frequently but uses fewer hash chains.

7 Discussion

Below, We will discuss key factors affecting AuthentiSafe's functionality, starting with limited authentication processes, PUF drift, large-scale networks, network disruptions, and device personalization. Additionally, we discuss accumulator selection, the storage system's role, and the impact of the Winternitz parameter in App. C.

Limited Authentication Keys. Potential concerns about the prover potentially exhausting its authentication keys, given that one key is invalidated in each time slot, are mitigated by the capabilities of modern PUF implementations. These typically offer 256-bit challenges and responses [78, 79], yielding potential of 2^{256} unique responses. This vast pool allows the prover to enroll a sufficient number of keys to last throughout the typical lifecycle of an ordinary IoT device. In our evaluation, we considered enrolling 2^{25} responses, resulting in 33,554,432 possible authentication processes. To put this into perspective, consider a standard 10-year lifecycle for an IoT device. With this number of enrolled responses, a device could perform over 9,000 authentications per day, or one every 10 seconds, which should be more than adequate for almost any scenario. Moreover, the computational effort for the IoT device only increases logarithmically with the tree height, allowing for the enrollment of even more PUF responses if necessary. This scalability ensures that the system can adapt to various operational requirements without significant performance degradation. This example underscores that, given the vast number of possible PUF responses and the ability to reuse each response multiple times through hash chains, the likelihood of a prover exhausting its authentication keys is extremely low. This holds true even under extended periods of inactivity or prolonged operation, providing a robust and long-lasting authentication solution for IoT devices.

PUF Drift. PUF drift [38] poses a potential risk to the reliability of AuthentiSafe. However, this challenge is inherent to all PUF-based systems, and several mitigation strategies have been proposed in response. These include drift-resistant PUF designs [58, 59], error correction techniques [65, 80], software-based approaches to counter PUF drift [38], and majority-voting mechanisms [64]. Each of these methods is aimed at enhancing the long-term reliability of PUFs and can be integrated with AuthentiSafe.

Large-Scale IoT Settings. AuthentiSafe is compatible with thousands of devices, as it utilizes only public information for authentication. As the network expands, the memory and computational demands for devices remain unchanged.

Disrupted Network Connection. Since all data required for a message from the prover to the verifier is locally available at the prover's end, the prover can prepare the message at any time and transmit it as soon as a network connection is available. Similarly, the verifier does not depend on a continuous connection to the prover for verification but only needs access to publicly available information. Therefore, in AuthentiSafe, network disruptions do not impact the overall flow of the protocol.

Device Personalization. To enhance the security of a device, classical authentication methods (e.g., passwords or biometric information) can be integrated into the protocol. The counter value can be combined with this information during both enrollment and authentication. This approach presents a potential extension for AuthentiSafe.

8 Related Work

Research on Physical Unclonable Function (PUF) based authentication falls into two categories: single-use and reuse strategies. Single-use methods generate unique challenge-response pairs (CRPs) for one-time authentication, while reuse methods employ the same pairs across multiple sessions.

Single-Use Approaches. PUF-based authentication strategies following the single-use approach have evolved along two primary paths. The first, exemplified in works such as [17, 19, 21, 44, 72, 73, 77]. These approaches involve generating an extensive challenge-response pair database during enrollment. Each authentication event utilizes a unique CRP, which is subsequently discarded. The second path leverages machine learning techniques. As demonstrated in [43, 56], these methods use the initial CRP set to train models capable of authenticating PUF responses without storing individual pairs. Despite their differences, both strategies impose significant confidential data storage demands on the verifier since they either require maintaining a large CRP database or necessitate storing complex predictive models. In both cases, the verifier must have significant secure storage capabilities making them unsuitable for resource-constraint IoT devices. In contrast, AuthentiSafe operates entirely on publicly available data generated during the enrollment phase, effectively shifting the storage burden from resource-limited IoT devices to public storage solutions, without the need for a trusted third party.

To mitigate storage challenges on resource-limited devices, researchers have explored alternative architectures. One approach, as seen in [9, 22, 23, 35], involves delegating PUF data management to a trusted third party. This entity securely stores confidential information on the verifier's behalf. Another strategy, exemplified in [2, 15, 18, 33, 36, 39, 45, 48, 81], introduces a trusted intermediary to facilitate secure communication between parties. These schemes either authenticate the prover on behalf of the verifier or establish an authenticated channel between devices. While innovative, these approaches still fall within the first category, as they merely shift the storage burden to a trusted third party rather than eliminating it. In contrast, AuthentiSafe eliminates the need for a trusted third party to store any information, as all data required for authentication is publicly available and outsourced to public storage solutions.

Notably, the work of Beckmann *et al.* [11] is different from those works. The authors propose Public PUFs (PPUFs), where each prover device has a public simulation of its internal PUF for verification. This eliminates the need to store confidential PUF responses.

However, the proposed simulator's complexity (requiring approximately 10^{13} clock cycles per verification) renders it impractical for resource-constrained IoT devices, which cannot efficiently perform such computationally intensive verifications. AuthentiSafe is significantly less computationally expensive and relies solely on publicly available information.

Multi-Use Approaches. This second category of systems utilizes PUF responses across multiple authentication procedures.

Zhang *et al.* [84] introduce a scheme using group signatures based on bilinear pairings and Merkle trees to enable anonymous authentication for IoT devices. In this approach, IoT devices must construct and securely store a Merkle tree for future authentication sessions. However, this is impractical for typical resource-constrained IoT devices due to their limited computational power and storage capacity. Additionally, the use of bilinear pairings during the authentication process exposes the protocol to vulnerabilities from post-quantum adversaries. In contrast, AuthentiSafe relies exclusively on post-quantum secure cryptography and offloads the storage of Merkle trees to public storage solutions, thereby avoiding additional computational or storage burdens on the IoT devices.

Felicetti *et al.* [31, 32] proposed combining PUFs with ECDSA [74] for secure digital signatures. Their approach uses PUF responses to generate ECDSA key pairs on demand, eliminating the need for persistent key storage. When a signature is required, the system queries the PUF, uses its response to derive an ECDSA key pair, and signs the message with the private key. Shariq *et al.* [68] followed the same approach by using the PUF response to generate a secret key on demand but used less common Schnorr signatures [66] instead of ECDSA. In both approaches, this method enhances security by avoiding long-term key storage and tightly coupling each signature to the device's unique physical characteristics. However, the approaches rely on computationally expensive public-key cryptography and lack post-quantum security, limiting future viability. In contrast, AuthentiSafe addresses these limitations, offering a more efficient and future-proof solution.

Delavar *et al.* [25] introduced an authentication scheme that enables the reuse of PUF responses usage of the Okamoto proof system [50]. This approach faces two significant drawbacks. Firstly, its reliance on complex cryptographic operations poses challenges for deployment on resource-limited IoT devices. Secondly, the protocol's design is limited to a single-verifier model, which fails to address the needs of modern IoT ecosystems where multiple stakeholders often require verification capabilities. Additionally, the Okamoto proof system is not post-quantum secure. In contrast, AuthentiSafe relies exclusively on lightweight post-quantum secure cryptographic primitives, making it both more efficient and future-proof.

Prada-Delgado *et al.* [55] propose an authentication scheme that integrates PUFs with the computational hardness of the xLPN problem [52]. While each individual round of this protocol is highly efficient, achieving adequate soundness requires over 100 iterations. This necessity leads to substantial communication overhead, potentially rendering the scheme impractical for resource-constrained IoT devices with limited bandwidth and power capabilities. In comparison, AuthentiSafe requires only a single protocol round for authentication, significantly reducing both the computational and communication burdens on the devices.

Petzi *et al.* [51] introduced a technique called challenge concealment combined with a zero-knowledge proof system for authentication, demonstrating knowledge of the correct PUF response without revealing it. However, the cryptographic operations of this approach are significantly more expensive compared to AuthentiSafe. Additionally, the employed cryptosystem lacks post-quantum security.

Samra *et al.* [63] proposed using the PUF response as a symmetric key to encrypt a pre-generated private elliptic curve key. During enrollment on the device, this private key is stored encrypted with the PUF response. For authentication, the PUF response is used to decrypt the private key, which is then employed for authentication purposes. While this approach avoids the need for regenerating the private key, as seen in [31] and [68], the utilized elliptic curve cryptography is significantly more computationally intensive than AuthentiSafe and lacks post-quantum security.

In summary, existing PUF-based authentication schemes typically fall into one of two categories: either they adopt a single-use approach, requiring large amounts of confidential storage to manage challenge-response pairs (CRPs) or machine learning models related to CRPs, or they reuse the same CRP multiple times to avoid dependence on confidential databases. AuthentiSafe introduces a novel solution by combining the advantages of both approaches. It employs a fresh CRP for each authentication process, preventing response reuse while eliminating the need for confidential information during verification. Additionally, AuthentiSafe is more efficient than previous schemes and stands out as the first PUF-based authentication protocol to offer post-quantum security. This makes it not only more secure but also highly suitable and future-proof for resource-constrained IoT devices.

9 Conclusion

Motivated by the need for a lightweight and future-proof authentication scheme for resource-constrained IoT devices, we propose AuthentiSafe, a PUF-based authentication scheme that blends single-use and multi-use approaches, inheriting the advantages of both. It utilizes PUFs to generate fresh authentication keys combined with the Winternitz one-time signature scheme to build a PUF-based authentication system based entirely on publicly verifiable information. Furthermore, the usage of a Merkle tree enables efficient verification even on resource-constrained devices. Compared to other PUF-based authentication schemes that require large databases or confidential information, AuthentiSafe offers faster execution times and is the first post-quantum secure scheme, as it solely relies on cryptographic hash functions, providing a future-proof alternative to currently existing solutions.

Acknowledgments

This research was funded by the European Union's Horizon Europe research and innovation program under grant agreement No 101070537, project CROSSCON (Cross-platform Open Security Stack for Connected Devices).

References

[1] Abidemi Emmanuel Adeniyi, Rasheed Gbenga Jimoh, and Joseph Bamidele Awotunde. 2024. A systematic review on elliptic curve cryptography algorithm for internet of things: Categorization, application areas, and security. *Computers and Electrical Engineering* (2024).

[2] Muhammad Naveed Aman, Kee Chaing Chua, and Biplab Sikdar. 2017. Physically Secure Mutual Authentication for IoT. *IEEE CDSC* (2017).

[3] Yang An, Yuejiao Zhang, Wenjun Cao, Zhiyan Tong, and Zhangqing He. 2022. A Lightweight and Practical Anonymous Authentication Protocol Based on Bit-Self-Test PUF. *Electronics* (2022).

[4] Manos Antonakakis, Tim April, Michael Bailey, Matt Bernhard, Elie Bursztein, Jaime Cochran, Zakir Durumeric, J Alex Halderman, Luca Invernizzi, Michalis Kallitsis, et al. 2017. Understanding the Mirai Botnet. *USENIX Security* (2017).

[5] Arm. 2023. Mbed TLS. https://github.com/Mbed-TLS/mbedtls. Accessed: 2024-03-28.

[6] Frederik Armknecht, Roel Maes, Ahmad-Reza Sadeghi, Berk Sunar, and Pim Tuyls. 2010. Memory Leakage-resilient Encryption based on Physically Unclonable Functions. *Towards Hardware-Intrinsic Security: Foundations and Practice* (2010).

[7] Gaurang Bansal, Naren Naren, and Vinay Chamola. 2020. RAMA: Real-Time Automobile Mutual Authentication Protocol Using PUF. *ICOIN* (2020).

[8] Gaurang Bansal, Naren Naren, Vinay Chamola, Biplab Sikdar, Neeraj Kumar, and Mohsen Guizani. 2020. Lightweight Mutual Authentication Protocol for V2G Using Physical Unclonable Function. *IEEE TVT* (2020).

[9] Mario Barbareschi, Alessandra De Benedictis, Erasmo La Montagna, Antonino Mazzeo, and Nicola Mazzocca. 2019. PUF-Enabled Authentication-as-a-Service in Fog-IoT Systems. *WETICE* (2019).

[10] Mario Barbareschi, Alessandra De Benedictis, and Nicola Mazzocca. 2018. A PUF-based hardware mutual authentication protocol. *J. Parallel and Distrib. Comput.* (2018).

[11] Nathan Beckmann and Miodrag Potkonjak. 2009. Hardware-Based Public-Key Cryptography with Public Physically Unclonable Functions. *Information Hiding* (2009).

[12] Oladayo Bello and Sherali Zeadally. 2014. Intelligent Device-to-Device Communication in the Internet of Things. *IEEE Systems Journal* (2014).

[13] Josh Benaloh and Michael De Mare. 1993. One-way Accumulators: A Decentralized Alternative to Digital Signatures. *EUROCRYPT* (1993).

[14] Juan Benet. 2014. IPFS - Content Addressed, Versioned, P2P File System. *arXiv preprint arXiv:1407.3561* (2014).

[15] An Braeken. 2018. PUF Based Authentication Protocol for IoT. *Symmetry* (2018).

[16] Johannes Buchmann, Erik Dahmen, Sarah Ereth, Andreas Hülsing, and Markus Rückert. 2013. On the security of the Winternitz one-time signature scheme. *International Journal of Applied Cryptography* (2013).

[17] Jin Wook Byun. 2019. An efficient multi-factor authenticated key exchange with physically unclonable function. *ICEIC* (2019).

[18] Urbi Chatterjee, Rajat Subhra Chakraborty, and Debdeep Mukhopadhyay. 2017. A PUF-Based Secure Communication Protocol for IoT. *TECS* (2017).

[19] Urbi Chatterjee, Rajat Sadhukhan, Vidya Govindan, Debdeep Mukhopadhyay, Rajat Subhra Chakraborty, Sweta Pati, Debashis Mahata, and Mukesh M Prabhu. 2018. PUFSSL: An OpenSSL Extension for PUF based Authentication. *DSP* (2018).

[20] Wenjie Che, Venkata K Kajuluri, Mitchell Martin, Fareena Saqib, and Jim Plusquellic. 2017. Analysis of Entropy in a Hardware-embedded Delay PUF. *Cryptography* (2017).

[21] Wenjie Che, Fareena Saqib, and Jim Plusquellic. 2015. PUF-Based Authentication. *ICCAD* (2015).

[22] Yun-Hsin Chuang and Chin-Laung Lei. 2021. PUF Based Authenticated Key Exchange Protocol for IoT Without Verifiers and Explicit CRPs. *IEEE Access* (2021).

[23] Vlastimil Clupek and Vaclav Zeman. 2016. Robust Mutual Authentication and Secure Transmission of Information on Low-cost Devices Using Physical Unclonable Functions and Hash Functions. *IEEE TSP* (2016).

[24] Li Da Xu, Wu He, and Shancang Li. 2014. Internet of Things in Industries: A Survey. *IEEE TII* (2014).

[25] Mahshid Delavar, Sattar Mirzakuchaki, Mohammad Hassan Ameri, and Javad Mohajeri. 2017. PUF-based solutions for secure communications in Advanced Metering Infrastructure (AMI). *International Journal of Communication Systems* (2017).

[26] Trinh Viet Doan, Yiannis Psaras, Jörg Ott, and Vaibhav Bajpai. 2022. Toward Decentralized Cloud Storage With IPFS: Opportunities, Challenges, and Future Considerations. *IEEE Internet Computing* (2022).

[27] Danny Dolev and Andrew Yao. 1983. On the Security of Public Key Protocols. *IEEE TIT* (1983).

[28] D Eastlake 3rd and Tony Hansen. 2011. *US Secure Hash Algorithms (SHA and SHA-based HMAC and HKDF).* Technical Report.

[29] Mohammad Ebrahimabadi, Mohamed Younis, and Naghmeh Karimi. 2021. A PUF-based Modeling-Attack Resilient Authentication Protocol for IoT devices. *IEEE Internet of Things Journal* (2021).

[30] Bin Fan, Dave G Andersen, Michael Kaminsky, and Michael D Mitzenmacher. 2014. Cuckoo Filter: Practically Better than Bloom. *CoNEXT* (2014).

[31] Carmelo Felicetti, Antonella Guzzo, Giuseppe Manco, Francesco Pasqua, Ettore Ritacco, Antonino Rullo, and Domenico Saccà. 2023. Deep Learning/PUF-based Item Identification for Supply Chain Management in a Distributed Ledger Framework. *BCCA* (2023).

[32] Carmelo Felicetti, Marco Lanuzza, Antonino Rullo, Domenico Saccà, and Felice Crupi. 2021. Exploiting Silicon Fingerprint for Device Authentication Using CMOS-PUF and ECC. *SmartIoT* (2021).

[33] Gurjot Singh Gaba, Mustapha Hedabou, Pardeep Kumar, An Braeken, Madhusanka Liyanage, and Mamoun Alazab. 2022. Zero Knowledge Proofs based Authenticated Key Agreement Protocol for Sustainable Healthcare. *Sustainable Cities and Society* (2022).

[34] Blaise Gassend, Dwaine Clarke, Marten Van Dijk, and Srinivas Devadas. 2002. Silicon Physical Random Functions. *CCS* (2002).

[35] Zhao Huang and Quan Wang. 2020. A PUF-based unified identity verification framework for secure IoT hardware via device authentication. *WWW* (2020).

[36] Qi Jiang, Xin Zhang, Ning Zhang, Youliang Tian, Xindi Ma, and Jianfeng Ma. 2019. Two-factor Authentication Protocol using Physical Unclonable Function for IoV. *2019 IEEE/CIC ICCC* (2019).

[37] Byoungkoo Kim, Seoungyong Yoon, Yousung Kang, and Dooho Choi. 2019. PUF based IoT Device Authentication Scheme. *IEEE ICTC* (2019).

[38] Michael S Kirkpatrick and Elisa Bertino. 2010. Software Techniques to Combat Drift in PUF-based Authentication Systems. *SECSI* (2010).

[39] JoonYoung Lee, JiHyeon Oh, DeokKyu Kwon, MyeongHyun Kim, SungJin Yu, Nam-Su Jho, and Youngho Park. 2022. PUFTAP-IoT: PUF-Based Three-Factor Authentication Protocol in IoT Environment Focused on Sensing Devices. *Sensors* (2022).

[40] Matt Lepinski and Stephen Kent. 2008. *RFC 5114: Additional Diffie-hellman groups for use with IETF standards.* Technical Report.

[41] Sensen Li, Tikui Zhang, Bin Yu, and Kuan He. 2020. A Provably Secure and Practical PUF-Based End-to-End Mutual Authentication and Key Exchange Protocol for IoT. *IEEE Sensors Journal* (2020).

[42] Wei Liang, Bo Liao, Jing Long, Yan Jiang, and Li Peng. 2016. Study on PUF based secure protection for IC design. *Microprocessors and Microsystems* (2016).

[43] Wei Liang, Songyou Xie, Dafang Zhang, Xiong Li, and Kuan-ching Li. 2021. A Mutual Security Authentication Method for RFID-PUF Circuit based on Deep Learning. *TOIT* (2021).

[44] Donglan Liu, Xin Liu, Hao Zhang, Hao Yu, Wenting Wang, Lei Ma, Jianfei Chen, and Dong Li. 2019. Research on End-to-End Security Authentication Protocol of NB-IoT for Smart Grid Based on Physical Unclonable Function. *ICCSN* (2019).

[45] Karim Lounis and Mohammad Zulkernine. 2021. T2T-MAP: A PUF-Based Thing-to-Thing Mutual Authentication Protocol for IoT. *IEEE Access* (2021).

[46] Thomas McGrath, Ibrahim E Bagci, Zhiming M Wang, Utz Roedig, and Robert J Young. 2019. A Puf Taxonomy. *Applied Physics Reviews* (2019).

[47] Ralph Charles Merkle. 1979. *Secrecy, authentication, and public key systems.* Stanford university.

[48] Muhammad Arif Muhal, Xiong Luo, Zahid Mahmood, and Ata Ullah. 2018. Physical Unclonable Function Based Authentication Scheme for Smart Devices in Internet of Things. *IEEE SmartIoT* (2018).

[49] Michael A Nielsen and Isaac L Chuang. 2001. *Quantum Computation and Quantum Information.* Vol. 2. Cambridge university press Cambridge.

[50] Tatsuaki Okamoto. 1992. Provably Secure and Practical Identification Schemes and Corresponding Signature Schemes. *CRYPTO* (1992).

[51] Lukas Petzi, Alexandra Dmitrienko, and Ivan Visconti. 2022. PUF-based Authentication in IoT against Strong Physical Adversary using Zero-Knowledge Proofs. *IEEE/ACM Workshop on the Internet of Safe Things* (2022).

[52] Krzysztof Pietrzak. 2012. Cryptography from Learning Parity with Noise. *SOFSEM* (2012).

[53] Thomas Pornin. 2013. *Deterministic usage of the digital signature algorithm (DSA) and elliptic curve digital signature algorithm (ECDSA).* Technical Report.

[54] Miodrag Potkonjak, Gang Qu, Farinaz Koushanfar, and Chip-Hong Chang. 2017. 20 Years of Research on Intellectual Property Protection. *ISCAS* (2017).

[55] Miguel Ángel Prada-Delgado, Iluminada Baturone, Gero Dittmann, Jens Jelitto, and Andreas Kind. 2020. PUF-derived IoT Identities in a Zero-Knowledge Protocol for Blockchain. *Internet of Things* (2020).

[56] Raffaele Pugliese, Stefano Regondi, and Riccardo Marini. 2021. Machine learning-based approach: Global trends, research directions, and regulatory standpoints. *Data Science and Management* (2021).

[57] Mahmood Azhar Qureshi and Arslan Munir. 2020. PUF-IPA: A PUF-based Identity Preserving Protocol for Internet of Things authentication. *CCNC* (2020).

[58] Md Tauhidur Rahman, Domenic Forte, Jim Fahrny, and Mohammad Tehranipoor. 2014. ARO-PUF: An aging-resistant ring oscillator PUF design. *DATE* (2014).

[59] Carl Riehm, Christoph Frisch, Florin Burcea, Matthias Hiller, Michael Pehl, and Ralf Brederlow. 2023. Structured Design and Evaluation of a Resistor-Based PUF Robust Against PVT-Variations. *DDECS* (2023).

[60] Ronald L Rivest, Adi Shamir, and Leonard Adleman. 1978. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM* (1978).
[61] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. 1978. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Commun. ACM* (1978).
[62] Ulrich Rührmair, Jan Sölter, and Frank Sehnke. 2009. On the Foundations of Physical Unclonable Functions. *Cryptology ePrint Archive, Paper 2009/277* (2009).
[63] SS Samra, KN Sreehari, and Ramesh Bhakthavatchalu. 2022. PUF Based Cryptographic Key Generation. *ASIANCON* (2022).
[64] Sudhir K Satpathy, Sanu K Mathew, Raghavan Kumar, Vikram Suresh, Mark A Anders, Himanshu Kaul, Amit Agarwal, Steven Hsu, Ram K Krishnamurthy, and Vivek De. 2019. An All-Digital Unified Physically Unclonable Function and True Random Number Generator Featuring Self-Calibrating Hierarchical Von Neumann Extraction in 14-nm Tri-gate CMOS. *IEEE JSSC* (2019).
[65] André Schaller, Taras Stanko, Boris Škorić, and Stefan Katzenbeisser. 2017. Eliminating Leakage in Reverse Fuzzy Extractors. *IEEE TIFS* (2017).
[66] Claus-Peter Schnorr. 1990. Efficient Identification and Signatures for Smart Cards. *CRYPTO* (1990).
[67] Alireza Shamsoshoara, Ashwija Korenda, Fatemeh Afghah, and Sherali Zeadally. 2020. A survey on physical unclonable function (PUF)-based security solutions forInternet of Things. *Computer Networks* (2020).
[68] Mohd Shariq, Karan Singh, Mohd Yazid Bajuri, Athanasios A Pantelous, Ali Ahmadian, and Mehdi Salimi. 2021. A Secure and Reliable RFID Authentication Protocol Using Digital Schnorr Cryptosystem for IoT-enabled Healthcare in COVID-19 scenario. *Sustainable Cities and Society* (2021).
[69] Peter W. Shor. 1997. Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. *SIAM J. Comput.* (1997).
[70] Nighat Solangi, Aiman Khan, Mehak Fatima Qureshi, Nafeesa Zaki, Umair Mujtaba Qureshi, and Zuneera Umair. 2024. IoT Based Home Automation System: Security Challenges and Solutions. *ICACS* (2024).
[71] Douglas R Stinson. 2005. *Cryptography: Theory and Practice.* Chapman and Hall/CRC.
[72] SD Suganthi, RVSS Anitha, Venkatasamy Sureshkumar, S Harish, and S Agalya. 2020. End to end light weight mutual authentication scheme in IoT-based healthcare environment. *Journal of Reliable Intelligent Environments* (2020).
[73] G Edward Suh and Srinivas Devadas. 2007. Physical Unclonable Functions for Device Authentication and Secret Key Generation. *DAC* (2007).
[74] O Sury. 2012. *RFC 6594: Use of the SHA-256 Algorithm with RSA, Digital Signature Algorithm (DSA), and Elliptic Curve DSA (ECDSA) in SSHFP Resource Records.* Technical Report.
[75] Vera Suryani, Selo, and Widyawan. 2016. ICITEE. *ICITEE* (2016).
[76] Pim Tuyls and Lejla Batina. 2006. RFID-tags for Anti-Counterfeiting. *Cryptographers' track at the RSA conference* (2006).
[77] Anthony Van Herrewege, Stefan Katzenbeisser, Roel Maes, Roel Peeters, Ahmad-Reza Sadeghi, Ingrid Verbauwhede, and Christian Wachsmann. 2012. Reverse Fuzzy Extractors: Enabling Lightweight Mutual Authentication for PUF-Enabled RFIDs. *Financial Cryptography and Data Security* (2012).
[78] Dai Yamamoto, Kazuo Sakiyama, Mitsugu Iwamoto, Kazuo Ohta, Masahiko Takenaka, and Kouichi Itoh. 2013. Variety Enhancement of PUF Responses using the Locations of Random Outputting RS Latches. *Journal of Cryptographic Engineering* (2013).
[79] Dai Yamamoto, Kazuo Sakiyama, Mitsugu Iwamoto, Kazuo Ohta, Masahiko Takenaka, Kouichi Itoh, and Naoya Torii. 2015. A New Method for Enhancing Variety and Maintaining Reliability of PUF Responses and its Evaluation on ASICs. *Journal of Cryptographic Engineering* (2015).
[80] Wei Yan, Fatemeh Tehranipoor, and John A Chandy. 2016. PUF-Based Fuzzy Authentication Without Error Correcting Codes. *IEEE TCAD* (2016).
[81] Seungyong Yoon, Byoungkoo Kim, Yousung Kang, and Dooho Choi. 2020. PUF-based Authentication Scheme for IoT Devices. (2020).
[82] Fahem Zerrouki, Samir Ouchani, and Hafida Bouarfa. 2022. PUF-based mutual authentication and session key establishment protocol for IoT devices. *Journal of Ambient Intelligence and Humanized Computing* (2022).
[83] Jiliang Zhang, Yaping Lin, Yongqiang Lyu, and Gang Qu. 2015. A PUF-FSM binding scheme for FPGA IP protection and pay-per-device licensing. *IEEE TIFS* (2015).
[84] Qingyang Zhang, Jing Wu, Hong Zhong, Debiao He, and Jie Cui. 2023. Efficient Anonymous Authentication Based on Physically Unclonable Function in Industrial Internet of Things. *IEEE TIFS* (2023).

A Visualization of Problem Scenario

In this paper, we consider large IoT networks with heterogeneous IoT devices from mutually mistrusting stakeholders. App. Fig. 5 shows an example of an IoT network with IoT devices that interact both within and across their stakeholders' boundaries to generate, collect, and share information.

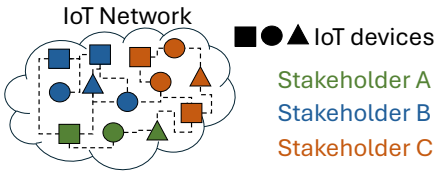

Figure 5: Multi-Stakeholder IoT Network.

B Additional Experimental Results

Below, we provide the results for the memory consumption experiments discussed in Sect. 6.3.

Table 4: Memory Consumption in Bytes.

LPC55S69-EVK board	Authentication		Verification		Sum
	Stack	Heap	Stack	Heap	
Felicetti *et al.* [31]	1,900	2,120	1,900	2,200	8,120
Samra *et al.* [63]	2,200	260	1,900	2,200	6,560
Petzi *et al.* [51]	328	1,900	328	1,900	4,456
AuthentiSafe ($w = 4$)	1,260	0	2,770	0	4,030
AuthentiSafe ($w = 8$)	688	0	1,730	0	2,418

C Additional Discussion

The following provides an extended discussion of our approach, serving as a supplementary addition to Sect. 7.

Influence of Winternitz Parameter. The total length of the signature is determined by the output length n of the used hash function and the Winternitz parameter w. Let n be the length of the used hash function (typically 256 bits for SHA-256). We first calculate $\ell_1 = \lceil \frac{n}{\log_2(w)} \rceil$, which represents the number of w-bit blocks needed to encode the message. We then determine $\ell_2 = \lfloor \frac{\log_2(\ell_1(w-1))}{\log_2(w)} \rfloor + 1$, representing the number of w-bit blocks needed to encode the checksum. The length $\ell = \ell_1 + \ell_2$ gives us the number of w-bit strings required for the WOTS scheme. Selecting the Winternitz parameter requires consideration of the system's requirements, including computational resources, bandwidth, and storage capacity.

Accumulator Selection. The accumulator used within AuthentiSafe is a Merkle tree [47], providing security based on cryptographic hash functions. While Merkle trees offer a robust and well-established solution, alternative data structures could be explored. An interesting alternative would be the use of probabilistic data structures, such as the cuckoo filter [30]. This data structure significantly reduces the amount of data to be stored while enabling even faster verification via membership tests compared to Merkle trees, with the slight trade-off of introducing a small probability of false positives. Although not suitable for high-risk environments where false positives are unacceptable, cuckoo filters might present a viable alternative for low-risk environments where devices are severely limited in terms of storage and computational resources.

File System used and its Necessity. To instantiate AuthentiSafe, we decided to store the individual terminal values on the InterPlanetary File System (IPFS) [14] for easy access during verification. However, other solutions can be used instead, such as cloud storage. Since the confidentiality of this data is not an issue, any storage solution that provides the necessary information availability during verification can be employed. The only necessity is that the system can provide the required Merkle path efficiently. An alternative approach to fully remove the public storage requirement would be to have the prover device generate the required values on-demand and transmit them together with the revealed chain element. This approach eliminates the need for public storage apart from the accumulator (or at least it's root node) but places additional computational overhead on the IoT device. As a result, this solution may be more suitable for more powerful IoT devices or devices equipped with hardware acceleration for cryptographic hash functions.