Future Generation Computer Systems 148 (2023) 425–435

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

# A lightweight remote attestation using PUFs and hash-based signatures for low-end IoT devices

Roberto Román *, Rosario Arjona, Iluminada Baturone

*Instituto de Microelectrónica de Sevilla, IMSE, CNM (Universidad de Sevilla, CSIC), C/Américo Vespucio 28, 41092, Seville, Spain*

ARTICLE INFO

*Article history:*
Received 15 December 2022
Received in revised form 9 May 2023
Accepted 6 June 2023
Available online 8 June 2023

*Keywords:*
Digital signatures
Hardware security
Internet of Things
Physically unclonable functions
Post-quantum cryptography
Security protocols

ABSTRACT

Remote attestation is a powerful mechanism that allows a verifier to know if the hardware of an IoT (Internet-of-Thing) device (acting as a prover) has been counterfeited or tampered with and if its firmware has been altered. Remote attestation is based on collecting and reporting measurements in a trusted way, and should be lightweight for resource-constrained IoT devices. This work proposes to include a low-cost Root of Trust for Measuring and Reporting (RoTMR) in the prover, based on the combination of a Physically Unclonable Function (PUF) and an Attestation Read-Only Memory (A-ROM), and to use hash-based digital signatures in the attestation protocol. The proposed RoTMR is addressed to IoT devices based on a microcontroller that executes some application code (the measurable object) located in an external non-volatile memory accessible by an attacker. The secret keys required by the digital signatures are not stored but reconstructed using the PUF. The A-ROM contains the attestation instructions and ensures that its contents cannot be altered and that its instructions are executed sequentially without modification. The use of hash-based digital signatures makes the solution quantum-resistant and very robust because its security relies solely on the unidirectionality of a hash function. The proposed attestation protocol takes advantage of the fact that One-Time Signature (OTS) generation and Many-Time Signature (MTS) verification are very well suited for low-end devices, and the MTS scheme is suitable for the verifier application context. The proposal was validated experimentally with the ESP32 microcontroller, which is widely employed in IoT devices, by using its SRAM as PUF and implementing WOTS+, which is a type of Winternitz One-Time Signature scheme (WOTS), the One-Time Signature of Smart Digital Signatures scheme (SDS-OTS), and the MTS schemes constructed with them. The OTS schemes require smaller codes and thus smaller A-ROM than MTS and ECDSA (Elliptic Curve Digital Signature Algorithm). The code of one of the WOTS+ takes about 4 times less space than ECDSA. In terms of execution times, the OTS schemes are very fast. One of the WOTS+ performs all the signature operations in a few tens of milliseconds. The OTS schemes (especially the SDS-OTS) are also very efficient in terms of communication bandwidth because they use small signatures compared to other post-quantum solutions.



## 1. Introduction

The Internet of Things (IoT) is a promising technology that has captured the attention of researchers in academia and industry around the world. The idea behind IoT is the interconnection of Internet-enabled things or devices to achieve some common goals. In the near future, IoT is expected to be seamlessly integrated into our environment and people will be completely dependent on it for a comfortable and easy lifestyle. Currently, IoT applications can be found in smart homes, wearable devices, smart grids, industry, healthcare, and agriculture, among others.

It is very important that the IoT provides persistent security. In fact, IoT security is a hot research topic today, mainly because of its heterogeneous nature. If an IoT device is compromised, it can cause serious damage where it is used. For example, a group of compromised IoT devices could form a botnet and launch a distributed Denial of Service (DoS) attack, or in the case of smart grid power, they could inject misinformation about usage into the control system, affecting the electrical network [1]. Since cyber–physical systems are interconnected, a threat to one device can spread to others, with catastrophic consequences. An attack at some point could have a direct impact on human lives. For this reason, ensuring the trustworthiness of IoT devices is a critical task. In particular, the security of low-end devices is more challenging because they are constrained in terms of power, computing, and memory resources.

* Corresponding author.
*E-mail addresses:* rrhajderek@us.es (R. Román), marjona@us.es (R. Arjona), lumi@us.es (I. Baturone).

https://doi.org/10.1016/j.future.2023.06.008

One way to know if a device is in a trusted state is through attestation [2]. An attestation scheme involves two mutually exclusive parties: a verifier V, and a prover P. In the context of IoT applications, the prover is an IoT device and the verifier is usually a remote device (which can be another IoT device or a server), which explains the name of remote attestation. Attestation is performed using a challenge-response mechanism. P performs a measurement of the device state to respond a challenge from V, V receives the measurement and then determines whether or not it represents a valid device state.

For a proper attestation operation, two main properties should be achieved [2]. The prover P should collect the measurements in a trusted way. This can be done by using a Root of Trust for Measuring and Reporting (RoTMR) integrated in the prover device [3]. In addition, the measurement should be unforgeable, that is, non-replayable and non-reconstructible so that an attacker should not be able to reproduce any measurement even if he knows the function used by P to compute the measurement or if the state of the IoT device has not changed. This can be achieved with the use of digital signatures.

Several taxonomies can be made depending on the form in which attestation is performed [4]. Hardware attestation focuses on validating the authenticity and trustworthiness of the IoT device hardware, for example, to detect if a counterfeit device is being used instead of the expected one, or if a genuine device has been tampered with. This type of attestation can be done by using Physically Unclonable Functions (PUFs), which allow measuring unique and intrinsic hardware properties of the IoT device and detecting if the device has been tampered with. In the software attestation, the measurement includes the application code in the device, i.e., the state reflects if the software or firmware has been altered or it is as expected.

This paper focuses on both hardware and software attestation for low-end IoT devices. Concerning hardware, a typical Root of Trust can be implemented using a Trusted Platform Module (TPM), which is a dedicated integrated chip with its own secure storage and execution unit. However, this solution is expensive for low-end IoT devices. This work proposes a low-cost RoTMR that uses an Attestation ROM (A-ROM) and a secret key obfuscated with a PUF. The secret key can be generated inside the device or can be provided externally in the registration phase. The proposed RoTMR is addressed to IoT devices based on a microcontroller executing some application code located in an external non-volatile memory. It is assumed that the microcontroller contains an A-ROM for the trusted execution of some specific firmware, such as the instructions dedicated to the attestation process. It is also assumed that a PUF is present in the hardware of the device. For software attestation, it is assumed that the application code is the measurable object and that it is located in a memory accessible by an attacker. The measurement is performed by applying a cryptographic hash function to the contents of the non-volatile memory.

In this work, hash-based signatures are explored for signing the memory measurements. Unlike the most commonly used digital signature algorithms, such as Rivest–Shamir–Adleman scheme (RSA) and Elliptic Curve Digital Signature Algorithm (ECDSA), hash-based signatures are post-quantum resistant, so no quantum computer can break them [5]. Unlike all other digital signatures found in literature, the security of hash-based signatures does not rely on any computationally hard problem, but solely on the unidirectionality of a hash function, making them very robust.

Hash-based signatures are hierarchical. At the core of any scheme is a One-Time Signature (OTS) scheme which can be used to sign only one message. On top of this are Many-Time Signature (MTS) schemes, which use a construction called Merkle tree to sign many messages using the same public key. One specified MTS scheme is XMSS (RFC8391) that uses an OTS scheme called WOTS+ [6]. More recently, an OTS scheme called SDS-OTS [7] was proposed, which achieves smaller signatures than the WOTS+. Among the MTS schemes, the XMSS and LMS (Leighton–Micali Signatures) are stateful hash-based signature standards, whose security depends on a careful state management so as to never reuse an OTS key. If an attacker were able to obtain two different messages signed with the same OTS key, he would be able to forge signatures on arbitrary messages [8]. Stateless hash-based signatures such as SPHINCS+ (which was selected for standardization after the third round of the NIST post-quantum cryptography standardization process [9]) do not require to keep track of any state between signatures. However, SPHINCS+ is not suitable for low-end devices. Therefore, this work proposes, in the one side, an efficient combination of stateful hash-based signatures (OTS and MTS schemes) because they are faster, produce smaller signatures, and require less attestation code than stateless schemes and, in the other side, a secure solution based on PUF and A-ROM that avoids key reusing.

The main contributions of this paper are the following:

- To establish a Root of Trust for Measuring and Reporting (RoTMR) based on the combination of an Attestation ROM (A-ROM) with a PUF. In this way, the prover device does not store the private keys employed in the one-time signatures, but reconstructs them from its PUF and never reuses them. Even in the context of simple authentication solutions with PUFs, this solution has not been well studied yet.
- To propose a quantum-resistant protocol for remote attestation using hash-based signatures, based on one-time signature generation and many-time signature verification. Both of these are not overly demanding operations, and, hence, they are well suited for low-end prover devices.
- To evaluate two different OTS schemes and an MTS scheme on an ESP32 microcontroller, which is widely employed in IoT devices. The chosen schemes are WOTS+ and the recent SDS-OTS, with their secret seeds recovered by an SRAM PUF from which different private keys are generated. Note that no previous work exists about the implementation of the SDS-OTS scheme in microcontrollers. In this work, the SDS-OTS proposed in [7] is implemented.
- To demonstrate the suitability of the proposal for IoT devices by measuring the sizes of the signatures and public keys to communicate, as well as, the execution times and code sizes of the relevant operations, which are mainly those related with cryptographic operations.

The paper is organized as follows. Section 2 presents some preliminaries needed to understand the proposal. Relevant works found in the literature are discussed in Section 3. The proposed solution for remote attestation is shown in Section 4 and experimental results are presented in Section 5 together with a comparison with other proposals. Finally, Section 6 concludes the paper.

## 2. Preliminaries

### 2.1. Physically obfuscating secrets with PUFs

A Physically Unclonable Function (PUF) can be defined as a function that is embedded into a physical object. When queried with a challenge, the PUF generates a response that depends on both the challenge and the unique device-specific properties of the physical object containing the PUF [10]. The most appealing PUFs for use in IoT devices are electronic PUFs where the physical object is an integrated circuit or part of it and the challenges and responses are represented as binary strings.

PUFs should satisfy robustness (when queried with the same challenge many times, the PUF returns similar responses with high probability), unclonability (it is unfeasible to produce two indistinguishable PUFs based on their challenge/response behavior), unpredictability (it is unfeasible to predict the response of a PUF to an unknown challenge, even if the PUF can be adaptively queried for a certain number of times) and tamper-resistance (the challenge/response behavior changes if the PUF is manipulated). PUFs are classified into weak and strong PUFs. Weak PUFs, such as Static Random-Access Memory (SRAM) PUFs and Ring Oscillator (RO) PUFs, provide a small number of challenge-response pairs compared to their number of basic constituent units (SRAM cells or ROs), while the strong PUFs, such as arbiter PUFs, provide a large number of challenge-response pairs [11]. SRAM PUFs are widely used in IoT devices because the SRAM is a constituent block of microcontrollers. In an SRAM PUF, binary challenges select a specific address range within the memory, and the responses are the power-up states of the cells at those addresses. The power-up states of SRAM cells can measure the random variability of the semiconductor manufacturing process.

The obfuscation and recovering of a secret with a PUF generally use a fuzzy extractor with a code-offset based secure-sketch [11]. In a generation phase, helper data $HD$ are computed from a PUF response $R$ and a secret $K$ generated by a True Random Number Generator (TRNG). The secret is conveniently encoded, $Enc(K)$, by using the encoder of an error-correcting code. The simplest generation of $HD$ uses XOR operations as follows $HD = R \oplus Enc(K)$. In a reproduction phase, another PUF response $R'$, which is similar to $R$ but not exactly the same, is employed to recover $R$ and the secret $K$ from $HD$, by using the decoder of the error-correcting code, $K = Dec(R' \oplus HD)$ and $R = HD \oplus Enc(K)$. Then, using a hash function, the recovered $R$ or $K$ is usually converted into a cryptographically secure and uniform random string to be used as a secret key [12]. An important property of fuzzy extractors is that the $HD$ can be stored and transmitted publicly without revealing the PUF response or the secret. An SRAM can be used to generate PUF responses and true random numbers for this application because SRAM cells can be classified into stable and random cells. If the cell shows preference to power up as 0 or 1, it is a stable cell that is good to be part of the PUF. If the cell shows no preference for any state, it is a random cell that is good to be part of a TRNG [13,14].

### 2.2. WOTS+

The WOTS+ scheme [6] is parameterized by the security parameter $n$ and by the Winternitz parameter $w$. Given them, the following sub-parameters are defined as:

$$t = \log_2(w),$$
$$len1 = \left\lceil \frac{n}{t} \right\rceil,$$
$$len2 = \left\lceil \frac{\lfloor \log_2(len1) \rfloor + 1 + t}{t} \right\rceil \text{ and}$$
$$len = len1 + len2$$

where $\lceil x \rceil$ is the ceil of $x$. The basic functions employed are a hash function $h$ and a $PRF$ (Pseudo Random Function), which is defined from $h$ as:

$$PRF\,(key, in) = h(key \,\|\, in)$$

where || is the concatenation operator.

The WOTS+ private key is composed of $len$ strings $sk(j)$ of size $n$, $SK_{OTS} = (sk\,(0) \,\|\, \ldots \,\|\, sk(len - 1))$. The strings are generated from a uniform random seed $SK_{SEED,OTS}$ as:

$$sk\,(j) = PRF(SK_{SEED,OTS}, j)$$

The WOTS+ public key is composed of $len$ strings $pk(j)$, $PK_{OTS} = (pk\,(0) \,\|\, \ldots \,\|\, pk\,(len - 1))$. The strings are computed by applying the hash function $w - 1$ times to each secret string $sk(j)$ as:

$$pk\,(j) = h^{w-1}(sk(j))$$

For the signature generation, the message digest $m_d$ (i.e. the result of the hash function applied to the message) is divided in $len1$ strings $m_{d,w}(j)$ of length $t$ as:

$$m_d = (m_{d,w}\,(0) \,\|\, \ldots \,\|\, m_{d,w}(len1 - 1))$$

Also, a checksum $C$ is computed as:

$$C = \sum_{j=0}^{len1-1} \left(w - 1 - m_{d,w}(j)\right)$$

and the result is encoded as a base-w value having as a result $len2$ strings $C_w\,(j)$ of size $t$ as:

$$C_w = (C_w\,(0) \,\|\, \ldots \,\|\, C_w(len2 - 1))$$

The concatenation of the $m_{d,w}(j)$ and $C_w(j)$ strings forms $len$ blocks $b(j)$. These blocks are used to create the signature strings as:

$$sig\,(j) = h^{b(j)}(sk(j))$$

so that the signature is:

$$Sig = (sig\,(0) \,\|\, \ldots \,\|\, sig(len - 1))$$

The verification of a signature $Sig' = (sig'\,(0) \,\|\, \ldots \,\|\, sig'(len - 1))$ of a message $m'$ is carried out by applying the hash function $w - 1 - b'(j)$ times to each signature string, where the blocks $b'(j)$ are obtained from $m'$ as explained for the blocks $b(j)$ in the signature generation. Verification is successful if the result matches the corresponding public key string as:

$$h^{w-1-b'(j)}\left(sig'\,(j)\right) = pk(j)$$

that is, if the public key is reconstructed from the received signature.

### 2.3. SDS-OTS

In the SDS-OTS scheme, which is presented in [7], there are exactly 17 secret key strings $sk\,(j)$ and each one has 384 bits, $SK_{OTS} = (sk\,(0) \,\|\, \ldots \,\|\, sk(16))$. The strings are generated from a uniformly random secret seed $SK_{SEED,OTS}$ by applying a hash function as:

$$sk\,(j) = h^{j+1}\left(SK_{SEED,OTS}\right) = h\,(sk(j - 1))$$

The public key $PK_{OTS}$ is also composed of 17 strings of 384 bits. The 16 first strings are generated by applying the hash function 96 times to each secret portion, and the last string, by applying the hash 1536 times. Note that the private and public keys have a fixed size of 816 bytes.

For the signature generation, the message digest $m_d$ is divided into 96 blocks $B\,(k)$ of 4 bits (i.e. 1 nibble). A mapping function $M\,(x)$ is defined that transforms a decimal input $x$ into the sum of its Tens and Units. For example, $M(16) = 1 + 6 = 7$ or $M(92) = 9 + 2 = 11$. Then, for every index $j$ (from 0 to 16), a set of numbers $s\,(r)$ is found, $S\,(j) = \{s\,(r)\}$, that are the results of the mapping function on the indexes $k$ of the blocks whose decimal value is equal to $j$. Once the sets $S\,(j)$ are specified, the function $y\,(m_d, j)$ is computed as:

$$y\,(m_d, j) = \left(\sum_{r=0}^{|S(j)|} s(r)\right) mod\ 96 + 1$$

for $j$ from 0 to 15. For $j = 16$, the function is computed in a checksum manner as

$$y(m_d, 16) = \sum_{j=0}^{15} (96 - y(m_d, j))$$

Then, the signature strings are computed as $sig(j) = h^{y(m_d,j)}(sk(j))$, so that the signature is $Sig = (sig(0) \| \ldots \| sig(16))$.

The signature verification computes the function $y(m'_d, j)$ from the digest of the received message $m'$, following the same steps as in the signature generation. Then, the public key strings are computed as:

$$pk'(j) = h^{96-y'(m_d,j)}(sig'(j))$$

for $j$ from 0 to 15. For $j = 16$, the corresponding string is computed as:

$$pk'(16) = h^{1536-y'(m_d,j)}(sig'(16))$$

The signature is validated if all the public key strings obtained are equal to the corresponding strings of the stored public key $PK_{OTS}$.

### 2.4. Many-time signatures: Merkle signature scheme (MSS) and XMSS

The OTS schemes described above are very lightweight but impractical for applications that require the signing of many messages and their validation by multiple parties. This is because many OTS public keys would have to be distributed among the parties. The Merkle Signature Scheme (MSS) was proposed to mitigate such limitation. The main idea is to use a Merkle tree, which is a complete binary tree, of height H to compact $2^H$ OTS public keys in only one, called the MSS public key, $PK_{MTS}$, whose size is equal to the size of the OTS public keys [15]. From the $2^H$ public–private key pairs generated by an OTS signature scheme, a Merkle tree is constructed where the leaves are the hashed public keys, the inner nodes are the hash of the concatenation of the child nodes in pairs, and the root node is the public key to be distributed. Using the same $PK_{MTS}$, $2^H$ messages can be signed.

The private keys are the $2^H$ OTS private keys. The $i$th OTS secret seed is generated as:

$$SK_{SEED,OTS}(i) = PRF(SK_{SEED}, i)$$

The signer must update securely the index $i$ every time an OTS signature $Sig(i)$ is generated. The verifier should know the index $i$ that indicates which of the $2^H$ secret keys was used for the OTS signature. Besides, the verifier needs the array of nodes required to reconstruct the $PK_{MTS}$ from the reconstructed OTS public keys. This array of nodes is called the authentication path, $Auth(i)$, and must be generated by the signer every time a message is signed. Fig. 1 illustrates a simple Merkle tree of height 3.

The MSS scheme can use directly the SDS-OTS scheme because the security of the SDS-OTS scheme relies on the collision resistance of the hash function used (the SHA384 is strongly recommend). In the case of the WOTS+ scheme, since its security is based on the second-preimage resistance of the hash function used, the Merkle tree must be randomized. This is why the eXtended Merkle Signature Scheme (XMSS) includes additional steps to make the MSS scheme with WOTS+ more robust. First, the leaves of the Merkle tree are not directly the hashes of the OTS public keys. Instead, $N$ Merkle trees called $L$-trees are used. The $i$th $L$-tree is a Merkle tree that uses the $len$ OTS public key strings of the $i$th OTS public key $PK_{OTS}(i)$ as leaves. Second, the message is not hashed alone to form the message digest. Instead, the message is concatenated with the XMSS public key $PK_{XMSS}$, the index $i$ and a nonce $R$ computed as:

$$R = PRF(SK_{PRF}, i)$$

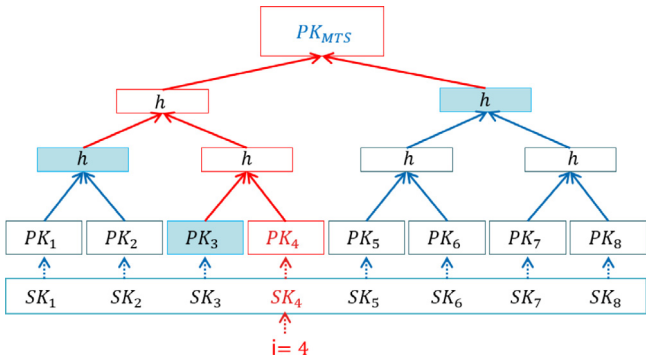


**Fig. 1.** Example of Merkle tree of height 3 with the authentication path of index 4 depicted in red.

**Table 1**
Number of hashes required for XMSS public key generation (PK gen.), signature generation (Sig. gen.) and signature verification (Sig. ver.) depending on the tree height H.

| No. hashes | | | |
|---|---|---|---|
| H | PK gen. | Sig. gen. | Sig. ver. |
| 10 | 123,016 | 5,725 | 1,149 |
| 16 | $79 \cdot 10^6$ | 9,163 | 1,155 |
| 20 | $1.268 \cdot 10^9$ | 11,455 | 1,159 |

The seed $SK_{PRF}$ must be uniformly random and must be kept secret. Finally, the most important difference in terms of security is that the hash functions are never used alone. Each time a hash function is called, the value $x$ to be hashed is XORed with a bitmask $bm_j$ and a key $k_j$ is appended as a prefix before the hash itself is applied. In the hash functions used for the $L$-trees and the main Merkle tree, two bitmasks are used (one for each child node). The keys and bitmasks should not be repeated, are public, and must be known by the verifier. Since they represent a large amount of data to be stored and transmitted, they are generated pseudo-randomly using a public seed $PUB_{SEED}$, a $PRF$ function and two address structures $ADRS_K$ and $ADRS_{bm}$, which change to ensure that no key or bitmask is repeated.

XMSS is more complex than MSS. In addition, the algorithms for public key generation, signature generation, and signature verification are very different. To illustrate this issue, Table 1 shows the number of hashes required for each algorithm in the XMSS scheme using three different tree heights [6]. Note that for typical heights of 16 and 20, signature verification requires almost an order of magnitude fewer hashes than signature generation. Note also that the number of hashes in the signature verification algorithm increases very slightly with the tree height.

## 3. Related work

In [1] two remote attestation protocols are presented. In the one aimed at resource-constrained devices, the prover must perform a one-time signature and generate two other one-time key pairs each time it is attested, and does not verify any digital signature. The protocol relies on a Trusted Third Party that verifies the prover's new one-time public key and sends it to the verifier. Both the verifier and the Trusted Third Party are allowed to use multi-time signatures. Another protocol without a Trusted Third Party is presented where the prover is assumed to be less resource-constrained and is allowed to perform multi-time signatures and verify the request of the verifier also with a multi-time scheme. The solution is verified using an Arduino 101 with a WiFi shield.

The work does not define a Root of Trust for Measurement (RoTM) or a Root of Trust for Reporting (RoTR) for the prover device.

In [16], the use of hash-based signatures in Trusted Platform Modules (TPMs) for signing attestation requests is proposed. The solution is intended for TPM-based solutions only. Typically, a TPM needs to manage multiple Attestation Identity Keys (AIK), and the use of MSS requires the storage of state information for each key in a trusted manner. Thus, the work presents a solution for managing this state information, given that a TPM has limited storage capabilities for storing information within the module. Although the solution is a step forward in the use of hash-based signatures for attestation purposes, it is not well suited to more constrained devices.

In [17], the authors integrate the quantum-resistant Crystals-Kyber and SPHINCS+ schemes into the open-source mbedTLS library to investigate how the TPM 2.0 specification can complement the migration towards post-quantum cryptography. The work is carried out in an Industrial Internet of Things (IIOT) environment and experimental results are presented using a Raspberry Pi 3 Model B with an Infineon SLB9760 discrete TPM. The TPM is only used to generate random seeds and accelerate hash operations, and the paper focuses mainly on mutual TLS (mTLS) authentication. The work does not investigate the attestation capabilities of the TPM.

Direct Anonymous Attestation (DAA) is a topic that is currently being studied in order to increase the privacy of devices and, consequently, the information that they manage. The work in [18] proposes an Enhanced Privacy ID (EPID), a DAA solution whose authentication technique is based on the group signature scheme of Boneh, Boyen, and Shacham and the group signature scheme of Furukawa and Imai. The work in [19] proposes a number of optimizations to EPID, mainly software-oriented but with the possibility of hardware acceleration. The authors argue that the optimizations make EPID suitable for constrained IoT devices and implement the solution in a 32 bit Intel Quark microcontroller. However, the latency using this form of attestation is high (17.9 s). In addition, EPID incorporates primitives that are not post-quantum resistant.

The use of hash-based signatures on constrained devices has also been studied in the literature. In [20], an area-latency-optimized hardware–software hybrid architecture is proposed to enable the XMSS scheme on resource-constrained IoT motes. For this, they use the Keccak-400 hash function (due to its lightweight nature), a set of small parameters, and perform design optimizations at the architecture level. Their target security is 128 bits. In [21], the authors provide a hardware–software co-design for the XMSS scheme on an embedded RISC-V processor. The proposal has speedups of 42× and 17× for signature generation and verification, respectively. The work in [22] shows the first secure boot using the XMSS scheme implemented in a RISC-V core. While it is true that the authors of all these works demonstrate that the use of XMSS is suitable for constrained IoT devices, significant hardware acceleration is needed to achieve low-latency signing times using the many-time scheme. Another work related to the migration of hardware primitives to the quantum world is [23]. It investigates the impact of using a Crystals-Dilithium signature verification in the secure boot of vehicle network processors.

In [24], the authors propose a new reputation and PUF-based remote identity attestation protocol for massive IoT devices in response to the security concerns in the identity attestation of massive IoT devices in smart cities. Three parties (aggregators, ordinary IoT devices, and Intelligent Operation Center (IOC)) interact in the protocol. Aggregators are the nodes with the highest reputation and help ordinary IoT devices to mutually authenticate with the IOC. The solution does not specify a specific Root of Trust to perform the attestation and focuses mainly on strong PUFs.

In [25], SIMPLE is proposed. It is a scheme that meets the minimal hardware requirements needed for remote attestation via trusted software. The proposal is based on a Security MicroVisor (S$\mu$V), which is a formally verified software-based memory isolation technique presented in [26]. The SIMPLE protocol uses a HMAC to authenticate the prover's digest to the verifier. Since HMAC is a symmetric cryptographic primitive, non-repudiation is not achieved, which can be a limitation in certain application contexts.

Several Roots of Trust other than the standard TPM solutions can be found in the literature for remote attestation, most of them targeting constrained devices. In [27], SMART is presented. It is a simple solution for establishing a dynamic Root of Trust for embedded devices. They use a minimal architecture which primarily relies on an attestation ROM and must guarantee key isolation, atomic execution and memory safety. Atomic execution means that instructions within the ROM memory cannot be executed in an isolated manner, and this is achieved by making minor changes to the hardware. The attestation key is stored inside the device and can only be accessed by the attestation ROM. Like other solutions focused on constrained devices, the prover uses an HMAC to authenticate to the verifier. Later, other solutions appeared, such as TrustLite [12], which added more features, such as support for interrupting tasks. More recent works use the idea of the attestation ROM with atomic execution in RISC-V based systems. In [28], LIRA-V is proposed. It is a lightweight system suitable for constrained devices that aims to establish a Root of Trust for remote attestation using the RISC-V architecture. They take advantage of the Physical Memory Protection (PMP) of the RISC-V and define a protocol where the prover device signs the attestation measurement with a digital signature. However, they do not consider quantum-secure primitives.

The combination of PUF solutions along with post-quantum cryptography and hardware security primitives is a recent line of research. In [13], the use of a SRAM-based PUF-TRNG is proposed to generate and obfuscate the secret seed used in the XMSS digital signature. However, like most works in the literature, no security is specified to prevent malicious access to the PUF, which is assumed to be secure. In [29], the first solution for secure sealed storage of sensitive data is proposed, combining a PUF with an atomic ROM and using quantum-resistant primitives, specifically Crystals-Dilithium and Saturnin. Three parties interact in the solution: a manufacturer, an application developer and the IoT device. In [30], the use of one-time hash-based digital signatures and SRAM PUFs is proposed for the secure boot of a microcontroller-based IoT device, achieving very low latency. No attestation protocol is proposed. In [31], the authors present a cryptosystem based on Hermitian curves suitable for IoT devices. The proposed algorithms perform encryption and decryption operations, but no authentication solution is presented.

## 4. The proposal for remote attestation

The proposal requires a Root of Trust in the IoT device that is described in Section 4.1. The security assumptions of our proposal are given in Section 4.2. The proposed protocol and its variants are presented in Section 4.3.

### 4.1. Device root of trust for measuring and reporting (RoTMR)

The proposed RoTMR has two main components, the *Attestation Read-Only Memory* (*A-ROM*) and the *Physically Unclonable Function* (*PUF*). These and other components such as the additional secure hardware *S-HW* are shown in Fig. 2.

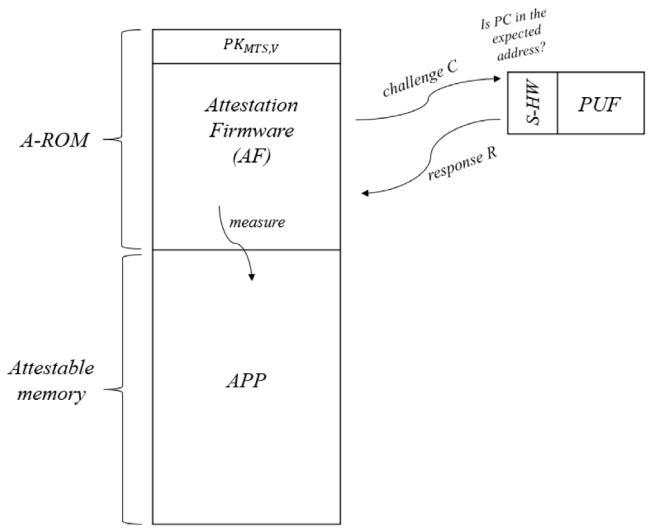


**Fig. 2.** Interaction between the RoTMR components: *Attestation Firmware*, the *PUF* and additional secure hardware *S-HW*, and the *Attestable memory*.

The *A-ROM* contains the *Attestation Firmware* (*AF*), which is the code that the prover must execute each time that the verifier requests an attestation. The region of addresses to be measured is referred to in this paper as the *Attestable memory*. The *A-ROM* must have two properties: (1) it must be *immutable*, i.e. its memory contents cannot be changed; and (2) it must be *atomic*, i.e. it must ensure that all the instructions are executed in a sequential order, and it must ensure the execution of each instruction. All the operations performed by the *Attestation Firmware* will be explained in the following sections. The immutable property is easily obtained by using a ROM. The public key of the verifier is included in the *A-ROM* since this key should also be immutable.

The atomic property of the *A-ROM* must be enforced by using the additional secure hardware *S-HW*. Different proposals for this additional hardware can be seen in [32] and in [33]. A useful approach is to validate the Program Counter (PC) with hardware. With this approach, if the Program Counter is an address within the *Attestation Firmware* other than the first instruction address, the *S-HW* should ensure that the previous instruction was also within the *Attestation Firmware*.

The *PUF* is used to obfuscate and recover the secret seed $SK_{SEED,OTS}$ used to sign the measurements in the OTS schemes. The *PUF* accepts a challenge *C* and returns a response *R*, both of which are processed by the *Attestation Firmware*. Using the *PUF*, the sensitive secret seed is not stored anywhere, but is recovered when needed, provided that the device is genuine and has a valid state. As mentioned in the previous sections, the use of *Helper Data HD* is necessary to obfuscate the secret seed. These Helper Data can be stored in a non-volatile memory as they are not sensitive data. It is very important that the *PUF* module is only accessed when it is executed by the *Attestation Firmware*. If this module is compromised, the trustworthiness of the attestation process is also compromised. This is again ensured by the *S-HW*, which can check whether the Program Counter is in the expected range of addresses, i.e. the addresses of the *Attestation Firmware*.

### 4.2. Assumptions

An attacker is assumed to be able to control all the writable memories of the prover device, and write and read any part of them for malicious purposes. Hence, an attacker can discover any secret if it is stored in plaintext. Since the *Application Code* can be stored in the non-volatile memory, its code execution flow can be modified by the attacker. Thus, he can have a persistent presence in the prover device. The *A-ROM* can be read by the attacker, but the functions stored in it cannot be modified and its execution cannot be altered.

An adversary is assumed to not perform sophisticated hardware attacks such as fault injection attacks, which could, for example, modify the value of the Program Counter or the received information conveniently signed by the verifier. These ones are assumed to be avoided by tamper-resistance techniques as mentioned in [27]. Also, the employed PUF is assumed to be resistant to side-channel attacks so that the attacker cannot read any physical response *R* from the PUF module and, hence, cannot obtain any information from the Helper Data and the secret is always obfuscated. The attacker may also have access to a quantum computer to break classical asymmetric cryptographic primitives such as ECDSA. Finally, concerning network capabilities, the adversary may have control over the communication channel between the prover device and the verifier.

### 4.3. Proposed protocol

The proposed protocol has two phases: enrollment and authentication. In the enrollment phase, the verifier *V* and the prover *P* exchange the necessary information to authenticate each other in the attestation process, which are the many-time public key of *V*, $PK_{MTS,V}$, and the first one-time public key of *P*, $PK_{OTS,P}(0)$. *V* also needs a reference memory measurement of *P*, $M_{MEM,GOLD}$. The enrollment should be done in a secure way. The manufacturer of the prover device could act as an intermediary between the two parties. In a simple scheme, *P* generates the secret seed $SK_{SEED,P}$ internally in the manufacturing facilities, obfuscates it having as a result the public Helper Data $HD_P$ that are stored into its non-volatile memory, and generates the first one-time public key $PK_{OTS,P}(0)$. Later, *P* sends the key $PK_{OTS,P}(0)$ to the manufacturer *M* and *M* certifies it by using its $SK_{MTS,M}$, thus resulting $CERT_M(PK_{OTS,P}(0))$. Finally, *M* sends the certificate $CERT_M(PK_{OTS,P}(0))$ and $PK_{OTS,P}(0)$ to *V* and grabs the verifier's many-time public key $PK_{MTS,V}$ in the prover device. It is assumed that *V* and *M* have the public keys of each other. This process is shown in Fig. 3. When the enrollment is finished, the manufacturer's role is also finished.

The authentication phase consists of 6 steps, as shown in Fig. 4. Each step is explained in the following:

**1.** Authentication starts with the verifier *V* signing a request *RQST* to the prover *P* to execute its Attestation Firmware and verify its state. *V* uses a Many-Time Signature (MTS) scheme like XMSS and must keep track of the number of its signatures with the index $i_V$. Since the prover's signatures must be synchronized, *V* must also keep track of the attestation index $i_{ATT}$. Furthermore, *V* updates its authentication path $AUTH_{MTS,V}(i_V)$ to be used in subsequent communications and stores it in its non-volatile memory. We assume that *V* can have multiple relationships with multiple devices, so $i_V$ does not necessarily match $i_{ATT}$. Both indexes $i_V$ and $i_{ATT}$ must be stored in a secure manner by the verifier, otherwise the security of the signatures may be compromised.

**2.** The verifier sends to the prover the request *RQST*, the attestation index $i_{ATT}$ and its signature $sig_{MTS,V}(i_V)$ along with the information needed to verify it, which is the verifier's signature index $i_V$ and the authentication path $AUTH_{MTS,V}(i_V)$.

**3.** The prover *P* executes its Attestation Firmware. The first thing that *P* does is to verify that the signature sent by the verifier *V* is correct. This verification is necessary because *P* will only proceed to execute its Attestation Firmware if the request comes from an authorized verifier. Since the signature scheme used by *V* is an MTS one, the verification process is divided into two steps: the

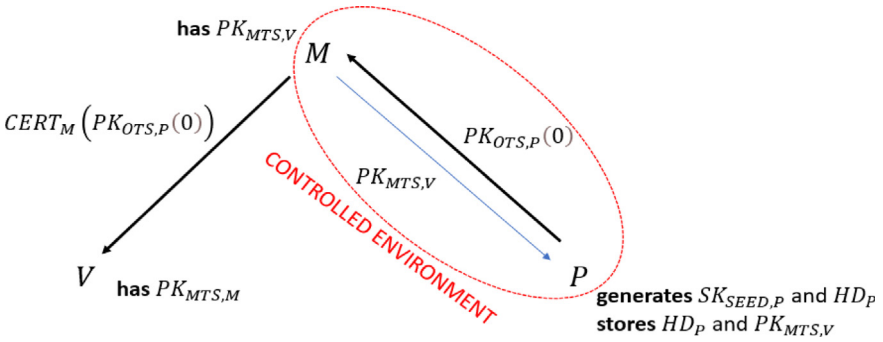


**Fig. 3.** Interactions between the verifier V, the prover P and the manufacturer M in the enrollment phase.

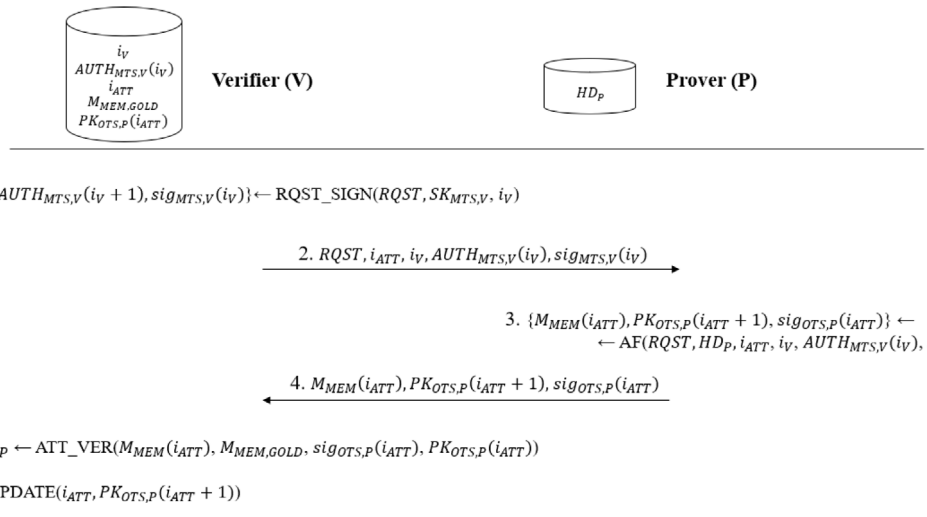


**Fig. 4.** Steps performed by the verifier and the prover in the authentication phase of the proposed protocol.

computation and the extraction of the root public key $PK_{MTS,V}$ using the array of authentication nodes specified in $AUTH_{MTS,V}$ $(i_V)$. If the signature is verified, the Attestation Firmware performs the measurement of the memory portions to be attested by applying a hash function to their contents, resulting in $M_{MEM}(i_{ATT})$. The memory portions were accorded with $V$ in the enrollment phase. Then, the obfuscated secret seed $SK_{SEED,P}$ is recovered with the Helper Data $HD_P$ stored in the non-volatile memory of the prover device and with a fresh PUF response $R_P$. Using a pseudo-random function ($PRF$), the secret seed $SK_{SEED,P}$ and the attestation index $i_{ATT}$ are mapped to $SK_{SEED,OTS,P}(i_{ATT})$. This seed is expanded to form the one-time private key $SK_{OTS,P}(i_{ATT})$, which is used to sign the memory measurement number $i_{ATT}$. Also, the secret seed $SK_{SEED,OTS,P}(i_{ATT}+1)$ is computed with the pseudo-random function $PRF$ and also expanded to $SK_{OTS,P}(i_{ATT}+1)$. Then, the OTS public key for the next attestation, $PK_{OTS,P}$ $(i_{ATT}+1)$, is computed using $SK_{OTS,P}(i_{ATT}+1)$, and the OTS key generation algorithm. Finally, $M_{MEM}(i_{ATT})$ is signed together with $PK_{OTS,P}(i_{ATT}+1)$ using the private key $SK_{OTS,P}(i_{ATT})$ and the OTS signature algorithm. Fig. 5 summarizes how the data involved in this step are generated.

**4.** The prover $P$ sends to the verifier $V$ the information that attests the state of its memory, which is the measurement $M_{MEM}$ $(i_{ATT})$, the OTS public key for the next attestation execution $PK_{OTS,P}$ $(i_{ATT}+1)$ and the signature of the concatenation of these two data $sig_{OTS,P}(i_{ATT})$. Note that the signature reflects the hardware state of $P$, since only the device with the correct PUF can recover the secret seed $SK_{SEED,OTS,P}(i_{ATT})$.

**5.** In this step the verifier $V$ checks if the state of the prover device is the expected one or if it has been compromised. The first step to do this is to verify if the OTS signature $sig_{OTS,P}(i_{ATT})$ made by the prover $P$ is correct by using the OTS public key $PK_{OTS,P}$ $(i_{ATT})$ and the OTS verification algorithm. If the OTS signature verification process is successful, it means that the hardware state of $P$ is correct. Then, the received measurement $M_{MEM}$ $(i_{ATT})$ is compared with the expected $M_{MEM,GOLD}$. If the two values are equal, the prover $P$ is considered to be in a trusted state.

**6.** At the end of the protocol, the verifier $V$ deletes the public key $PK_{OTS,P}$ $(i_{ATT})$, which is replaced by $PK_{OTS,P}$ $(i_{ATT}+1)$, and updates the attestation index from $i_{ATT}$ to $i_{ATT}+1$. Since the verifier is assumed to be trusted and an attacker cannot modify the index received by the prover or forge the verifier's signature, the prover never uses the same private key for two different messages.

## 5. Experimental results and discussion

In order to evaluate and validate the proposal, the selected IoT Platform acting as prover device was the Pycom WiPy 3.0 Development Board [34] based on the Espressif ESP32 microcontroller [35]. Specifically, the ESP32D0WDQ6 version was used. The board is equipped with an 8 MB SPI Flash memory for non-volatile storage outside the microcontroller. The ESP32 used can support applications with WiFi and Bluetooth connectivity since it comes with an integrated antenna switch, RF balun, power amplifier, low-noise receive amplifier, filters and power management modules. The external SMD antenna included in the WiPy 3.0 board makes this development board a good candidate for IoT devices.

The processor used by the ESP32 chip is the 32 bit dual-core Xtensa LX6 with a 16/24 bit instruction set. The available CPU frequencies are 80, 160 and 240 MHz synthesized by a 40 MHz

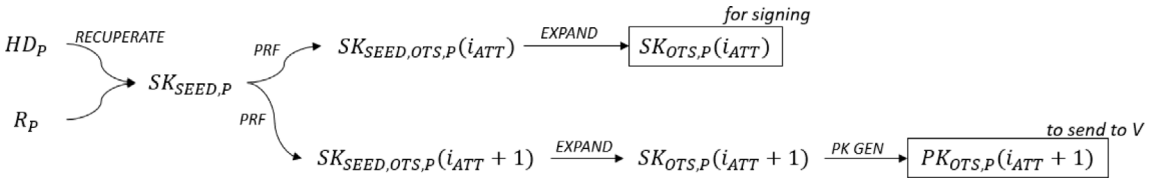


**Fig. 5.** Generation of the data involved in step 3 of the authentication phase.

**Table 2**
Signature and public key sizes for WOTS+ and SDS-OTS schemes.

| Scheme | WOTS+ ($w = 4$) | WOTS+ ($w = 16$) | WOTS+ ($w = 256$) | SDS-OTS |
|---|---|---|---|---|
| Signature size (bytes) | 4256 | 2144 | 1088 | 816 |
| Public Key size (bytes) | 32 | 32 | 32 | 48 |

**Table 3**
Execution times related to signature operations of WOTS+ and SDS-OTS schemes.

| Execution times in ms | | | | |
|---|---|---|---|---|
| Scheme | WOTS+ ($w = 4$) | WOTS+ ($w = 16$) | WOTS+ ($w = 256$) | SDS-OTS |
| PK gen. | 64.81 | 149.93 | 1265.33 | 109.53 |
| Sig. gen | 32.96 | 66.78 | 594.36 | 55.38 |
| Sig. ver. | 44.80 | 96.03 | 684.01 | 65.99 |
| Total | 142.57 | 312.74 | 2543.70 | 230.90 |

crystal oscillator. It also has on-chip SRAM with a size of 520 kB. The WiPy 3.0 board incorporates a 4 MB pSRAM but it was not used to execute the *Attestation Firmware* because the on-chip memory was sufficient for this purpose. The on-chip SRAM of the ESP32 was used as an SRAM PUF as described and evaluated in [13] and [36]. The ESP32 has support for Random Number Generation (RNG) using noise captured at the antenna, but this functionality was not used since the start-up values of the random cells of the SRAM were used to generate the secret seed required by the OTS schemes.

The SHA hash function was used to hash the messages and also to generate and verify the signatures. The SHA256 and SHA384 implementations found in the mbedtls library were used for SHA computations in software. The SHA accelerator available in the ESP32 microcontroller was used to accelerate the hash operations. The hwcrypto library was used for hardware acceleration. The achieved speedup for WOTS+ schemes was of $\times 2.15$ but the most benefited was the SDS-OTS scheme, with a speedup of $\times 4$.

The main aspects considered in this study were the execution times of the different operations explained in Section 5, and the sizes of the signatures, the public keys and the codes, since they are related to the power consumption and memory resources of the prover device. The execution times were obtained by reading the CCOUNT register of the Xtensa LX5 processor, which returns the cycle count of the CPU clock. A comparison using the two OTS schemes explained in Section 2 is offered to know which one may be more suitable in the context of device attestation. Following the RFC8391 specification [6], the three possible Winternitz parameters ($w$ equal to 4, 16, and 256) and a security of 256 bits were chosen for the WOTS+ scheme. As shown in Section 2, the Winternitz parameter establishes a trade-off between the size of the signature/keys and the latency in the execution of the algorithms [6].

The signature sizes using the WOTS+ and SDS-OTS schemes are shown in Table 2. The SDS-OTS signature scheme provided the smallest signature size and, therefore, the least amount of communication bandwidth is consumed by this scheme. This is particularly important for hash-based signatures since they have relatively large signatures. The WOTS+ scheme manages larger signatures, which vary depending on the Winternitz parameter chosen. As the Winternitz parameter increases, the signature size decreases, with the largest being 4256 bytes and the smallest being 1088 bytes for Winternitz parameters of 4 and 256, respectively. The Winternitz parameter of 16 had a moderately large signature size (2144 bytes). Table 2 also shows the size of the one-time public keys. The technique used was that reported in [37], which replaces the L-trees with simple hash calls. Thus, the public keys were more compact and smaller in size.

Execution times are primarily dominated by the OTS signature generation, the MTS signature verification, and the public key generation, as these are clearly the most expensive operations of the *Attestation Firmware* for a moderate size of the *Attestable memory*.

Table 3 shows the execution timing results for the different OTS scheme variants. The message used for signing and verification was "Good message 707070". The CPU frequency used was 160 MHz. The Merkle tree used was of height H = 20. The seed expansion was included in the public key generation, "PK gen.", and the signature generation, "Sig. gen.", as well, as the hash operation to transform the $SK_{SEED,P}$ into the $SK_{SEED,OTS,P}(i_{ATT})$. Note that the total execution time of "Sig. ver." + "PK gen." + "Sig. gen." includes all major operations of the *Attestation Firmware* except for the memory measurement and the PUF-related operations. The authentication path verification is included in the "Sig. ver." operation. Table 3 shows that the public key generation is the most expensive operation among the cryptographic operations of the *Attestation Firmware*. The table also shows that the SDS-OTS scheme reported shorter times in all the operations compared to WOTS+ ($w = 16$). WOTS+ ($w = 4$) is significantly faster than the other schemes with the disadvantage of larger signature sizes, as shown in Table 2. It is interesting to note that the verification time of an ECDSA (secp256r1) in this ESP32 was 203.74 ms, which means that the MTS schemes with SDS-OTS, WOTS+ ($w = 4$) and WOTS+ ($w = 16$) are faster.

Compared to the execution times shown in Table 3, the time to recover the secret seed SK_SEED is very small, being of 2.02 ms for WOTS+ and 3.03 for SDS-OTS, as reported in [36]. The difference between the schemes is due to the use of a seed of 32 bytes in WOTS+ and a seed of 48 bytes in SDS-OTS. The error correcting code used was an 8 bit repetition correcting code and the Helper Data Algorithm was a simple XOR operation, as explained in Section 2. The time required to measure a 4 kB sector of the non-volatile memory (*Attestable memory*) was of 1.73 ms for WOTS+ and 1.70 ms for SDS-OTS. Note that SDS-OTS uses the SHA384 hash algorithm instead of SHA256 (SHA384 processes fewer data chunks than SHA256).

Regarding code size, the measurements considered the main cryptographic operations required by the protocol. They are shown in Fig. 6. Since the code size of the WOTS+ scheme does not depend much on the Winternitz parameter, only the case with w

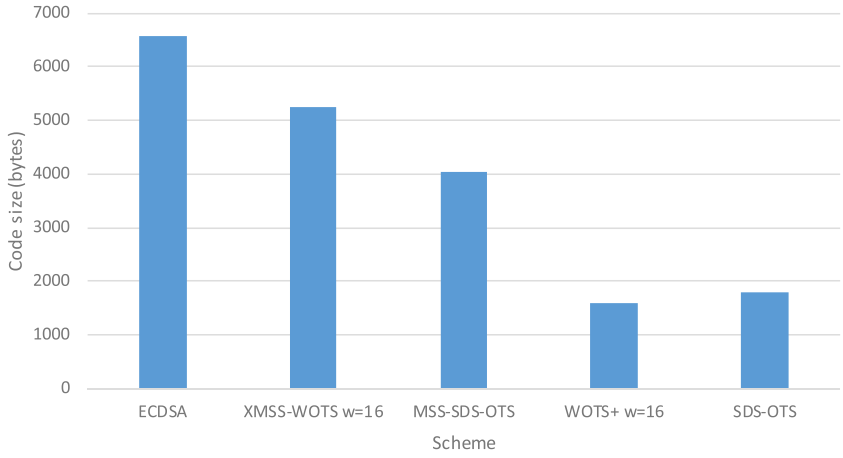


**Fig. 6.** Code size that must be stored for OTS operations along with the ECDSA case and XMSS for WOTS+ and MSS for SDS-OTS cases in bytes.

**Table 4**
Comparison with related work on remote attestation.

| Proposal | RoTMR | PUF | A-ROM | Post-Quantum signatures | Low-end devices |
|---|---|---|---|---|---|
| [1] | No | No | No | Yes | Yes |
| [16] | Yes | No | No | Yes | No |
| [25] | No | No | No | No | Yes |
| [27] | Yes | No | Yes | No | Yes |
| [12] | Yes | No | Yes | No | Yes |
| This work | Yes | Yes | Yes | Yes | Yes |

$= 16$ is shown, together with the SDS-OTS scheme. In the case of WOTS+ and SDS-OTS, the code for the one-time public key generation, signature generation and the verification are considered. In the case of ECDSA, XMSS and MSS, only the signature generation and verification operations are considered (because it is assumed that no public key needs to be generated). The code for the WOTS+ scheme was about 4 times smaller than ECDSA. The XMSS scheme had a code size approximately 3.3 times larger than WOTS+. Similar results were found for SDS-OTS. It should be emphasized that the code size of WOTS+ and SDS-OTS is dominated by the hash operation, being 56% of the former and 49% of the latter. Thus, our proposal requires a smaller *Attestation ROM* memory. In addition, the use of the SHA accelerator allowed a 23.69% reduction in the code size of the WOTS+ variant and a 21.33% reduction in the code size of the SDS-OTS variant. Code sizes were extracted from .map files.

Considering the proposal and the experimental results, some points can be discussed:

- *More flexibility using WOTS+.* The fact that the WOTS+ specified in [6] allows the use of three different parameters, more flexibility is achieved in the proposal. If a very fast response from the prover device is needed, WOTS+ ($w = 4$) can be used, and if a small bandwidth is needed, WOTS+ ($w = 256$) can be used. WOTS+ ($w = 16$) is the best compromise. Note that WOTS+ with w $= 4$ has a large signature size compared to the other variants, but it has a signature size even smaller than other post-quantum digital signatures such as Crystals-Dilithium, which has a signature size of 4595 bytes.
- *Advantages of SDS-OTS.* The SDS-OTS scheme has smaller signatures and is the proposal with the lowest bandwidth. It also has speed advantages over WOTS+ ($w = 16$) if the hash function is hardware accelerated since SHA384 processes fewer data chunks than SHA256. However, if a 32 bit processor is used, and no hardware acceleration is employed, WOTS+ ($w = 4$) and WOTS+ ($w = 16$) are faster than SDS-OTS because SHA256 processes 32 bit chunks. Note that the results shown in Table 3 use the SHA accelerator. However, SDS-OTS is a less studied scheme in the literature and may not fit future standards for post-quantum cryptography.
- *Smaller code sizes for OTS schemes.* The OTS algorithms (especially the WOTS+ with $w = 16$) require smaller code sizes and, hence, smaller A-ROM than MTS and ECDSA.

As can be seen in Table 4, which compares our proposal with the related work summarized in Section 3, our proposal differs from the others mainly by combining lightweight primitives such as a PUF, an A-ROM, and post-quantum OTS schemes.

## 6. Conclusions

This work proposes a lightweight remote attestation protocol for low-end IoT devices. It is based on the use of a low-cost Root of Trust for Measuring and Reporting (RoTMR) included in the IoT device acting as prover. The main components of the RoTMR are a Physically Unclonable Function (PUF) and an Attestation Read-Only Memory (A-ROM). The PUF avoids the storage of secret keys while the A-ROM allows attestation instructions to be immutable and executed in an atomic way. Hash-based signatures, which are quantum-resistant, are employed in the protocol. The prover device generates one-time signatures (OTS) and verifies many-time signatures (MTS) sent by the verifier.

To evaluate the proposal, the Pycom WiPy 3.0 Development Board, which includes an ESP32 microcontroller, was employed as IoT device. The on-chip SRAM in the microcontroller was used as PUF and as True Random Number Generator (TRNG) to generate the secret seed needed by the OTS schemes. Two OTS schemes (WOTS+ and SDS-OTS) and the MTS constructed with them were implemented in the ESP32. Parameters were selected to achieve a post-quantum security of 256 bits. Results were obtained using the hardware accelerator incorporated in the ESP32 microcontroller for the hash operations.

Although WOTS+ achieves more compact and smaller sizes than SDS-OTS in terms of public key sizes, the SDS-OTS scheme is the best in terms of communication bandwidth because it uses signatures with the smallest sizes. In terms of code sizes of the main operations, the results are similar for WOTS+ and SDS-OTS. Both use less code than ECDSA and XMSS. In terms of execution times, WOTS+ with w $= 4$ is the fastest option.

Future work is planned to evaluate the impact of using the standardized SPHINCS+ in the signature verification step of the protocol.

## CRediT authorship contribution statement

**Roberto Román:** Conceptualization, Methodology, Software, Investigation, Data curation, Writing – original draft, Visualization, Supervision, Funding acquisition. **Rosario Arjona:** Methodology, Investigation, Data curation, Writing – original draft, Visualization, Supervision, Project administration, Funding acquisition. **Iluminada Baturone:** Conceptualization, Methodology, Investigation, Data curation, Writing – original draft, Visualization, Supervision, Project administration, Funding acquisition.

## Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Data availability

No data was used for the research described in the article

## Acknowledgments

This research was conducted thanks to Grant PDC2021-121589-I00 funded by MCIN/AEI/10.13039/501100011033, Spain and the "European Union NextGenerationEU/PRTR, Spain", and Grant PID2020-119397RB-I00 funded by MCIN/AEI/10.13039/5011000 11033, Spain. The work of Roberto Román was supported by VI Plan Propio de Investigación y Transferencia, Spain through the Universidad de Sevilla.

## References

[1] X. Liu, R. Misoczki, M.R. Sastry, Remote attestation for low-end prover devices with post-quantum capabilities, in: Proceedings of the Eighth ACM Conference on Data and Application Security and Privacy, CODASPY '18, ACM, New York, New York, USA, 2018, pp. 84–94, http://dx.doi.org/10.1145/3176258.3176324.

[2] O. Arias, et al., Device attestation: Past, present, and future, in: 2018 Design, Automation & Test in Europe Conference & Exhibition, DATE, 2018, pp. 473–478, http://dx.doi.org/10.23919/DATE.2018.8342055, https://ieeexplore.ieee.org/document/8342055/authors.

[3] Global Platform Technology, Root of Trust Definitions and Requirements Version 1.1, 2018, https://globalplatform.org/wp-content/uploads/2018/07/GP_RoT_Definitions_and_Requirements_v1.1_PublicRelease-2018-06-28.pdf.

[4] R. Vieira Steiner, E. Lupu, Attestation in wireless sensor networks: A survey, ACM Comput. Surv. 49 (3) (2016) 1–31, http://dx.doi.org/10.1145/2988546.

[5] A. Hülsing, et al., Hash-based signatures: An outline for a new standard, in: Workshop on Cybersecurity in a Post-Quantum World, 2015, https://csrc.nist.gov/csrc/media/events/workshop-on-cybersecurity-in-a-post-quantum-world/documents/papers/session5-hulsing-paper.pdf.

[6] A. Hülsing, et al., RFC 8391: XMSS: EXtended Merkle Signature Scheme, IETF, 2018, https://www.rfc-editor.org/rfc/rfc8391.

[7] F. Shahid, A. Khan, Smart digital signatures (SDS): A post-quantum digital signature scheme for distributed ledgers, Future Gener. Comput. Syst. 111 (2020) 241–253, http://dx.doi.org/10.1016/j.future.2020.04.042, https://www.sciencedirect.com/science/article/pii/S0167739X19319892?via%3Dihub.

[8] L. Groot Bruinderink, A. Hülsing, Oops, I did it again – Security of one-time signatures under two-message attacks, 2016, Cryptology ePrint Archive, IACR https://eprint.iacr.org/2016/1042.pdf.

[9] CSRC-NIST, Post-quantum Cryptography Round 3 Submissions. https://csrc.nist.gov/Projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-3-submissions.

[10] R. Maes, I. Verbauwhede, Physically unclonable functions: A study on the state of the art and future research directions, towards hardware-intrinsic security, Inf. Secur. Cryptogr. (2010) http://dx.doi.org/10.1007/978-3-642-14452-3_1.

[11] Y. Gao, et al., Physically unclonable functions, Nat. Electron. 3 (2020) 81–91, http://dx.doi.org/10.1038/s41928-020-0372-5.

[12] P. Koeberl, et al., TrustLite: A security architecture for tiny embedded devices, in: Proceedings of the Ninth European Conference on Computer Systems, EuroSys '14, ACM, New York, New York, USA, 2014, http://dx.doi.org/10.1145/2592798.2592824.

[13] R. Román, et al., Hardware security for extended Merkle signature scheme using SRAM-based PUFs and TRNGs, in: 32nd International Conference on Microelectronics, ICM, 2020, pp. 1–4, http://dx.doi.org/10.1109/ICM50269.2020.9331821.

[14] I. Baturone, et al., Improved generation of identifiers, secret keys, and random numbers from SRAM, IEEE Trans. Inf. Forensics Secur. 10 (12) (2015) http://dx.doi.org/10.1109/TIFS.2015.2471279, https://ieeexplore.ieee.org/document/7217837.

[15] J. Buchmann, E. Dahmen, A. Hülsing, XMSS - A practical forward secure signature scheme based on minimal security assumptions, in: B.-Y. Yang (Ed.), Post-Quantum Cryptography, Springer Berlin Heidelberg, Berlin, Heidelberg, 2011, pp. 117–129, http://dx.doi.org/10.1007/978-3-642-25405-5_8.

[16] M. Ando, et al., Hash-based TPM signatures for the quantum world, in: Applied Cryptography and Network Security, ACNS, Springer, 2016, http://dx.doi.org/10.1007/978-3-319-39555-5_5.

[17] S. Paul, et al., TPM-based post-quantum cryptography: A case study on quantum-resistant and mutually authenticated TLS for IoT environments, in: Proceedings of the 16th International Conference on Availability, Reliability and Security, ARES 21, ACM, New York, New York, USA, 2021, http://dx.doi.org/10.1145/3465481.3465747.

[18] E. Brickell, J. Li, Enhanced privacy id: A direct anonymous attestation scheme with enhanced revocation capabilities, IEEE Trans. Dependable Secure Comput. 9 (3) (2011) 345–360, http://dx.doi.org/10.1109/TDSC.2011.63, https://ieeexplore.ieee.org/document/6109283.

[19] S. Ghosh, et al., Anonymous attestation for IoT, 2019, Cryptology ePrint Archive, IACR https://eprint.iacr.org/2019/121.pdf.

[20] S. Ghosh, et al., Lightweight post-quantum-secure digital signature approach for IoT motes, 2019, Cryptology ePrint Archive, IACR https://eprint.iacr.org/2019/122.

[21] W. Wang, et al., XMSS and embedded systems - XMSS hardware accelerators for RISC-V, 2018, Cryptology ePrint Archive, IACR https://eprint.iacr.org/2018/1225.pdf.

[22] V.B.Y. Kumar, et al., Post-quantum secure boot, in: 2020 Design, Automation & Test in Europe Conference & Exhibition, DATE, pp. 1582–1585, http://dx.doi.org/10.23919/DATE48585.2020.9116252.

[23] J.W. Bos, et al., Post-quantum secure boot on vehicle network processors, 2022, Cryptology ePrint Archive, IACR https://eprint.iacr.org/2022/635.

[24] J. Cao, et al., RPRIA: Reputation and PUF-based remote identity attestation protocol for massive IoT devices, IEEE Internet Things J. 9 (19) (2022) 19174–19187, https://ieeexplore.ieee.org/document/9749059.

[25] M. Ammar, et al., SIMPLE: A remote attestation approach for resource-constrained IoT devices, in: 2020 ACM/IEEE 11th International Conference on Cyber-Physical Systems, ICCPS, 2020, pp. 247–258, http://dx.doi.org/10.1109/ICCPS48487.2020.00036, https://ieeexplore.ieee.org/abstract/document/9096052.

[26] W. Daniels, et al., S$\mu$V - the security microvisor: A virtualisation-based security middleware for the internet of things, in: Proceedings of the 18th ACM/IFIP/USENIX Middleware Conference: Industrial Track, Middleware '17, ACM, New York, New York, USA, 2017, https://dl.acm.org/doi/abs/10.1145/3154448.3154454.

[27] K. El Defrawy, et al., Smart: Secure and minimal architecture for (establishing a dynamic) root of trust, in: Network and Distributed System Security Symposium, 2012, https://www.ics.uci.edu/~gts/paps/smart.pdf.

[28] C. Shepherd, et al., LIRA-V: Lightweight remote attestation for constrained RISC-V devices, 2022, http://dx.doi.org/10.48550/arXiv.2102.08804, arXiv.

[29] R. Román, I. Baturone, Sealed storage for low-cost IoT devices: An approach using SRAM PUFs and post-quantum cryptography, in: European Interdisciplinary Cybersecurity Conference, EICC, ACM, New York, New York, USA, 2021, pp. 54–59, http://dx.doi.org/10.1145/2592798.2592824.

[30] R. Román, I. Baturone, A quantum-resistant and fast secure boot for IoT devices using hash-based signatures and SRAM PUFs, in: The Fifth International Conference on Safety and Security with IoT, Springer, 2022, http://dx.doi.org/10.1007/978-3-030-94285-4_8.

[31] O.A. Alzubi, J.A. Alzubi, O. Dorgham, M. Alsayyed, Cryptosystem design based on Hermitian curves for IoT security, J. Supercomput. 76 (2020) 8566–8589, http://dx.doi.org/10.1007/s11227-020-03144-x.

[32] A. Francillon, et al., A minimalist approach to remote attestation, in: 2014 Design, Automation & Test in Europe Conference & Exhibition, DATE, 2014, pp. 1–6, http://dx.doi.org/10.7873/DATE.2014.257, https://ieeexplore.ieee.org/document/6800458.

[33] X. Carpent, et al., Reconciling remote attestation and safety-critical operation on simple IoT devices, in: 55th ACM/ESDA/IEEE Design Automation Conference, DAC, 2018, pp. 1–6, http://dx.doi.org/10.1109/DAC.2018.8465928.2.
[34] WiPy Datasheet Version 1.0, Pycom, 2018, https://pycom.io/wp-content/uploads/2018/08/Pycom_Specsheet_WiPy3.0.pdf.
[35] ESP32 technical reference manual version 4.7, 2022, Espressif Systems https://www.espressif.com/sites/default/files/documentation/esp32_technical_reference_manual_en.pdf.
[36] J. Arcenegui, et al., Secure combination of iot and blockchain by physically binding iot devices to smart non-fungible tokens using pufs, Sensors 21 (9) (2021) http://dx.doi.org/10.3390/s21093119.
[37] F. Campos, et al., LMS vs XMSS: Comparison of stateful hash-based signature schemes on ARM Cortex-M4, 2020, Cryptology ePrint Archive, IACR https://eprint.iacr.org/2020/470.

**Roberto Román** received the bachelor's degree in Electronic Engineering of Telecommunications from the University of Barcelona in 2018 and the Master of Science in Microelectronics from the University of Seville in 2020. He is currently pursuing a Ph.D. in Microelectronics at the Microelectronics Institute of Seville (IMSE-CNM), University of Seville, CSIC. In 2019 he won a JAEIntro grant of CSIC and now his research is supported by the VI Plan Propio de Investigación y Transferencia of the University of Seville. His current research interests are post-quantum cryptography, blockchain technologies, self-sovereign identities, and biometrics.

**Rosario Arjona** received the degree in computer science engineering and the Ph.D. degree (Hons.) in microelectronics from the University of Seville, Spain, in 2009 and 2014, respectively. Since September 2009, she has been with the Instituto de Microelectrónica de Sevilla, University of Seville-CSIC. Currently, she is an Assistant Professor at the Department of Electronics and Electromagnetism, University of Seville. She has collaborated with 16 national and international research and industrial projects. She is author of 32 scientific articles and one patent. Her main research interests include crypto-biometrics, Physical Unclonable Functions (PUFs), blockchain, Internet of Things (IoT), and neuro-fuzzy systems.

**Iluminada Baturone** received the 5-year B.S. degree (Hons.) in physics and the Ph.D. degree (Hons.) in physics-electronics from the University of Seville, Seville, Spain, in 1991 and 1996, respectively. She is Full Professor at the University of Seville with the Department of Electronics and Electromagnetism and Senior Researcher at the Microelectronics Institute of Seville (IMSE-CNM), University of Seville, CSIC. She has coauthored 2 books and more than 150 papers. She has filed 4 patents and participated in more than 40 projects, leading 13 of them. Her current research interests include hardware security, post-quantum cryptography, biometrics, blockchain technologies, and neuro-fuzzy systems.