PERMANENT: Publicly Verifiable Remote Attestation for Internet of Things through Blockchain

Sigurd Frej Joel Jørgensen Ankergård, Edlira Dushku ✉ [0000−0002−4974−9739], and Nicola Dragoni[0000−0001−9575−2990]

DTU Compute, Technical University of Denmark (DTU), Kgs. Lyngby 2800, Denmark
s164443@student.dtu.dk, {edldu,ndra}@dtu.dk

Abstract. Remote Attestation (RA) is a security mechanism that allows a centralized trusted entity (Verifier) to check the trustworthiness of a potentially compromised IoT device (Prover). With the tsunami of interconnected IoT devices, the advancement of *swarm* RA schemes that efficiently attest large IoT networks has become crucial. Recent swarm RA approaches work towards distributing the attestation verification from a centralized Verifier to many Verifiers. However, the assumption of trusted Verifiers in the swarm is not practical in large networks. In addition, the state-of-the-art RA schemes do not establish network-wide decentralized trust among the interacting devices in the swarm. This paper proposes PERMANENT, a Publicly Verifiable Remote Attestation protocol for Internet of Things through Blockchain, which stores the historical attestation results of all devices in a blockchain and allows each interacting device to obtain the attestation result. PERMANENT enables devices to make a trust decision based on the historical attestation results. This feature allows the interaction among trustworthy devices (or with a trust score over a certain threshold) without the computational overhead of attesting every participating device before each interaction. We validate PERMANENT with a proof-of-concept implementation, using Hyperledger Sawtooth as the underlying blockchain. The conducted experiments confirm the feasibility of the PERMANENT protocol.

Keywords: Remote attestation; Internet of Things; Blockchain; Public verifiability

1 Introduction

With the rapid evolution of the Internet-of-Things (IoT), many smart devices are increasingly becoming interconnected, working together in remote locations and performing many collaborative tasks without human intervention. Often IoT devices perform safety-critical operations and process sensitive information, thus, these devices are continuously targeted from many cyber attacks [25, 30, 17, 19]. While it is challenging to protect resource-constrained devices with conventional security mechanisms, Remote Attestation (RA) has emerged as a lightweight security method that verifies whether devices have been compromised or not. Traditional RA is a challenge-response protocol between a trusted party called Verifier (*Vrf*) and an untrusted remote device called Prover (*Prv*). Specifically, at the attestation time, the *Vrf* sends a challenge to the *Prv*, the *Prv* responds by sending reliable evidence about its software state to the *Vrf*. This evidence consists of performing a software measurement (i.e., computing a checksum or hash) usually over the static memory content of a device which allows the *Vrf* to detect the malware presence on *Prv*'s device. However, traditional challenge-response RA protocols pose scalability challenges for large IoT systems [3].

In order to overcome the scalability challenges of RA, many swarm RA schemes ([8], [5], [4], to mention only a few) have been proposed in the literature to allow a trusted *Vrf* to attest large-scale IoT networks. The state-of-the-art RA approaches typically rely on the presence of multiple Verifiers in the swarm for verification. Nevertheless, the assumption of trusted Verifiers is often not practical in large networks. Moreover, the existing RA schemes do not establish network-wide decentralized trust among the interacting devices in the swarm.

Contribution of the Paper. We argue that, to establish trust in multi-party large IoT networks, blockchain is a promising technology [14]. In particular, the immutability of blockchain guarantees the reliability of IoT data stored in blockchain transactions. Moreover, all the historical transactions stored in the blockchain are traceable. While these properties are important in improving IoT data security in general, they can potentially play a key role in securing attestation evidence of IoT devices.

To the best of our knowledge, this paper proposes the first RA protocol that uses blockchain technology to make RA publicly verifiable. Specifically, instead of relying on any single trusted third party, we rely on permissioned blockchains [18] to establish trust in a decentralized manner. In our approach, devices perform self-attestation [22], [11], [27] which gets triggered by a timer stored in the device's trusted component. Then, we leverage the timer of Proof-of-Elapsed-Time (PoET) consensus mechanism to combine it with the device's timer used for self-attestation in order to reach consensus without additional interactions. The proposed protocol utilizes the blockchain-based history of the devices attestation to evaluate the trustworthiness of IoT devices. The paper brings the following two main contributions:

1. The paper designs PERMANENT, a novel RA protocol which leverages blockchain technology to make the attestation result publicly verifiable and decentralized. PERMANENT decides devices' trustworthiness based on their entire historical attestations evidence. This feature serves as a building block to enable secure interactions among IoT devices.
2. The paper presents the proof-of-concept implementation of PERMANENT. PERMANENT has been implemented and tested with HyperLedger Sawtooth using Proof-of-Elapsed-Time (PoET) as a consensus mechanism. Experiments confirm the feasibility of the proposed solution.

Outline. The remainder of this paper is organized as follows. Section 2 presents different RA approaches and compares PERMANENT with relevant ones. System model and adversary model are described in Section 3 and Section 4, respectively. Next, PERMANENT protocol is detailed in Section 5 and its proof-of-concept implementation presented in Section 6. Protocol limitations are discussed in Section 7. Finally, Section 8 concludes the paper.

2 Related works

In general RA is classified into three categories: software-based, hardware-based and hybrid RA. Software RA [29], [6] does not require specialized hardware components but instead uses timing requirements to ensure the attestation code has not been tampered with. However, software-based RA schemes rely on strong adversarial assumptions and do not provide secure storage for protecting device's keys and the attestation code. To tackle this drawback, hardware-based RA relies on specialized hardware components like Trusted Platform Module (TPM) [7] to provide a root-of-trust. TPM consists of a coprocessor that performs software measurements during system boot and securely stores RA cryptographic keys. However, such a specialized hardware component for RA is expensive and not practical for IoT devices. Hybrid RA [16], [10] relies only on minimal additional hardware components, such as Read-Only Memory (ROM) and memory protection unit (MPU). The hardware components of hybrid approaches are cheaper, making them more suitable for an IoT setting. Thus, the current state-of-the-art RA protocols are based on hybrid architecture.

Self-triggering RA. Instead of following a classical on-demand challenge-response protocol, self-attestation schemes self-trigger the attestation based on a timer resided in a trusted component. **SEED** [22] is a non-interactive RA protocol where the RA time is determined from a pseudo-random number generator (PRNG), for which both the *Prv* and the *Vrf* have the seed. Once the timer is triggered, the *Prv* performs RA. Then, the *Prv* uses the shared symmetric key to sign the RA result along with the RA time so that the *Vrf* can check the *Prv*'s trustworthiness and RA freshness. **ERASMUS** [11] is a RA protocol that aims to solve the problem of on-demand RA requiring a device to stop normal operations to perform RA. In ERASMUS, the *Prv* uses a reliable read-only clock to perform RA at pre-defined times. The *Prv* then stores the RA results locally in its memory, and the *Vrf* can collect a set of consecutive RA results. In this way, the *Vrf* can identify a mobile adversary that tries to hide itself during RA.

Swarm RA. Swarm RA schemes (e.g., [8], [5], [4], [3]) focus on attesting a group of devices efficiently. **SEDA** [8] constructs the network as a spanning tree to allow efficient propagation of RA request and aggregation of the RA responses. The aggregated RA result is then sent to a centralized trusted *Vrf*. **SANA** [5] extends SEDA by employing a multi-signature scheme that aggregates the RA results among a large group of devices. The usage of multi-signature makes the RA publicly verifiable in SANA because anyone who knows that public key can

Table 1. Remote Attestation schemes using Blockchain

RA scheme	**Public/Private**	**Consensus**	**Blockchain**	**RA**	**Decentralized**
BARRET	Public	PoW	Ethereum	Any	No
TM-COIN	Public	PoW	Own	Hardware	No
DAN	Private	PBFT	HyperLedger Fabric	Hardware	No
PERMANENT	**Private**	**PoET**	**HyperLedger Sawtooth**	**Hybrid**	**Yes**

verify the aggregated RA result. In general, swarm RA schemes are on-demand protocols initiated by a trusted *Vrf*.

Distributed RA. Distributed services RA schemes (e.g., [12], [13], [15]) aim to attest a group of interacting devices that compose a *distributed IoT service*. **RADIS** [13] performs control-flow RA of synchronous distributed services by representing the entire control-flow execution of a distributed service as a single hash value. **SARA** [15] attests asynchronous distributed IoT services in a publish/subscribe IoT network. Both RADIS and SARA attest distributed services while relying on the presence of a centralized trusted *Vrf*. Instead, the distributed RA schemes (e.g., [2], [21], [24]) overcome the need for a centralized trusted *Vrf*, e.g., a base station, to handle RA. In particular, devices in the network play the role of the *Prv* and the *Vrf*. As such, devices in the network attest each other. **DIAT** [2] performs control-flow RA for each pair of devices. In **US-AID** [21], devices perform mutual attestations and store the result of their neighbour to assess the health status of the entire network. In **ESDRA** [24], each *Prv* gets attested by three different neighbours that assign a score to the *Prv*. In the end, the *Prv*'s score is reported to cluster-heads and then to the *Vrf*. In distributed RA schemes, the verification process is distributed across many Verifiers, but the RA results are not publicly verifiable.

2.1 Remote attestation using Blockchain

BARRET [9] aims to mitigate computational Denial of Service attacks by utilizing an Ethereum blockchain. It works by forcing the *Vrf* to pay a computational fee to send a RA request, which is the fee for mining a blockchain block. Since Verifiers have to pay this fee, they cannot send thousands of (valid) RA requests to a *Prv*. In BARRET, a *Vrf* sends a RA request to the blockchain, and the blockchain smart contract forwards this RA request to the *Prv*. Once the *Prv* receives the request, it performs RA, submits the result to the blockchain, and sends it to the *Vrf*. Then, the *Vrf* checks and submits the verification result to the blockchain. **TM-COIN** [26] is a hardware-based RA scheme utilizing blockchain to store the RA results. Here, a *Vrf* challenges a *Prv*, and the *Prv* stores the evidence in the blockchain. At any time, the *Vrf* can check the blockchain to see if a *Prv* is trustworthy. TM-COIN uses its own blockchain architecture, a public blockchain with Proof-of-Work (PoW) consensus algorithm. However, it is not a completely decentralized system since the miners are still

Table 2. Overview of Consensus Algorithms efficiency

Algorithm	Family	Throughput	Scalability	Overhead
Proof-of-Work (PoW)	Proof-of-X	Low	Low	Computational
Proof-of-Authority (PoA)	Proof-of-X	Low	High	None
Proof-of-Stake (PoS)	Proof-of-X	Low	Low	None
Proof-of-Elapsed-Time (PoET)	**Proof-of-X**	**Low**	**High**	**None**
Proof-of-Capacity (PoC)	Proof-of-X	Low	Low	None
Proof-of-Burn (PoB)	Proof-of-X	Low	Low	None
Proof-of-Importance (PoI)	Proof-of-X	Low	Low	None
Byzantine Fault Tolerance (BFT)	Voting	High	Low	Communications
Crash Fault Tolerance (CFT)	Voting	High	High	Communications

responsible for performing the PoW and verifying the RA response. **DAN** [23] is a hardware-based RA scheme that uses a Trusted Platform Module (TPM) as a root-of-trust. It clusters devices into organizations where each organization has a number of peer nodes responsible for interactions with the blockchain. Here, a *Vrf* sends a challenge to the device and waits for the RA result. The peer node is responsible for adding the result to the blockchain. In DAN, the proof-of-concept implementation relies on HyperLedger Fabric and the peer nodes are containers running on consumer desktops.

Discussion. Table 1 shows an overview of the three RA schemes utilizing blockchain technology. It shows that two of them, BARRET and TM-COIN, use public blockchains with PoW consensus algorithm. In contrast, DAN uses a private blockchain with a Practical Byzantine Fault Tolerance (PBFT) consensus. Furthermore, TM-COIN and DAN rely on specialized hardware components, while BARRET abstracts away from the RA and hardware requirements. All three schemes rely on trusted Verifiers to verify RA response and/or super nodes to handle blockchain interactions. Thus, they are not completely decentralized. Different from the existing blockchain-based RA schemes, PERMANENT aims to provide a decentralized RA using a Hyperledger Sawtooth as a permissioned blockchain with Proof-of-Elapsed-Time (PoET) as a consensus mechanism.

2.2 Blockchain Consensus Protocols

In designing a blockchain network, the choice of the consensus protocol is crucial mainly due to its significant impact on performance. Table 2 presents an overview of the consensus algorithms efficiency. While voting-based consensus protocols provide a better performance, they introduce a communications overhead, which is costly in an IoT environment. The PoX category has two protocols, PoET and PoA, which both have high scalability, but they have a low throughput (Transactions per second), which makes them poor choices if there are many transactions to be added to the blockchain. However, in the RA context, the throughput is a low priority metric because RA does not occur very often. To this end, PoET is a suitable consensus protocol with good performance, offering both low computational and low communications overhead. In PoET consensus, each network participant is given a random timer and the participant that has the shortest time (the timer that expires first) becomes the block leader and produces the new block. Thus, PoET brings an advantage in our proposed RA protocol: We leverage PoET's timer to combine it with the *Prv*'s timer (protected by the trusted component) used for self-attestation. In this way, when the timer triggers RA, it also allows the device to add a new block in the blockchain with the corresponding RA result. Additionally, in comparing different blockchains architectures, the study in [28] shows that HyperLedger Sawtooth clearly outperforms other HyperLedger blockchains in an IoT setting. Thus, in this paper, we choose HyperLedger Sawtooth with PoET consensus algorithm.

3 System model

We consider a peer-to-peer (P2P) IoT network where untrusted IoT devices interact among themselves. In such a system, each device must be authenticated in order to join the permissioned blockchain network that uses Proof-of-Elapsed-Time (PoET) consensus mechanism. Devices that are part of the network have permission to add blocks in the blockchain. Unauthenticated entities have only read permissions to the blockchain data. Each participating IoT device acts both as a Prover (*Prv*) and Verifier (*Vrf*). Note that we assume that devices are trusted in the beginning when they authenticate to join the blockchain network (e.g., they can be enforced to perform attestation), but they can be compromised later, so in general we consider a network of untrusted devices.

We assume the presence of a Network Operator (OP) that guarantees the secure bootstrap of RA protocol and blockchain code deployed on each device. OP computes the checksum (i.e., collision-resistant hash) of the device's legitimate software and stores the corresponding valid measurement inside each device. In addition, the OP ensures secure key distribution among devices.

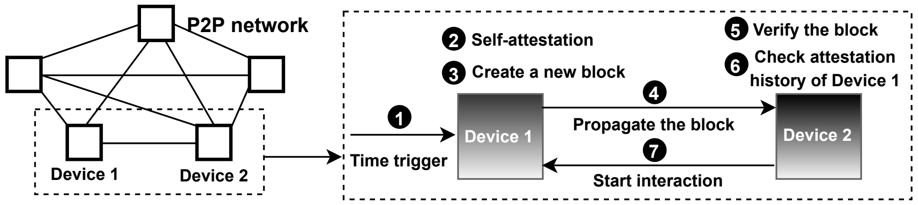

Fig. 1. Overview of interactions between two devices in the blockchain network

We consider the interactions among untrusted devices in a P2P blockchain network, and for simplicity, Figure 1 depicts the interactions among two devices in the network. The RA procedure starts when the timer of Device 1 gets triggered (Step ❶). Then, Device 1 performs self-attestation by computing the software measurement and comparing the computed result against the pre-stored legitimate attestation value (1 if it matches and 0 otherwise) (Step ❷), and adds the boolean attestation result in a new blockchain block (Step ❸). When the new block is published to the blockchain, it is broadcasted and propagated throughout the network (Step ❹). When the peers (e.g., Device 2 in Figure 1) receive the new block, they verify it by checking the device's signature and the signed timer (Step ❺). Later, when Device 2 wants to communicate to another device in the network (e.g., Device 1), it first checks the blockchain for the historical results of the device's attestation and then decides its trustworthiness (Step ❻). If the device is trustworthy above a pre-defined threshold, then these two devices proceed with their interaction (Step ❼).

4 Adversary model and Security requirements

4.1 Adversary model

In line with the adversary model described in [1], [3], and in particular, with other swarm and self-attestation schemes (e.g., [22], [11], [8], [5]), we consider an adversary with the following capabilities.

- **Software adversary (Adv_{sw}):** A Adv_{sw} exploits a vulnerability on *Prv*'s software and compromises the *Prv* by executing malicious code.
- **Communication adversary (Adv_{comm}):** The Adv_{comm} can forge, drop, delay, and eavesdrop the exchanged messages among devices.
- **Mobile adversary (Adv_{mob}):** A Adv_{mob} tries to avoid detection by deleting itself just before the execution of the attestation protocol starts.
- **Replay attack:** Any of the adversaries above can precompute a valid attestation response and responds with the old valid attestation response to hide malware presence.

Assumptions. We assume that a Adv_{sw} does not compromise the hardware-protected memory. Following the assumptions of other RA schemes [22], [11], we rule out physical adversaries. While we do not consider Denial of Service (DoS) and Distributed Denial of Service (DDoS) attacks, we limit these attacks by relying on self-attestation approach where the attestation request is not initiated by the *Vrf*. In addition, the current scope of the paper does not consider attacks that directly target the blockchain.

Device requirements. In line with common assumptions of the state-of-the-art RA schemes (e.g., [22], [15], [20], [20], [31]), we assume the presence of three trusted components inside a *Prv*.

- **Read-Only Memory (ROM).** The code of PERMANENT protocol and blockchain reside in a ROM memory region, preventing software adversaries Adv_{sw} from tampering with the code.

- **Secure key storage.** This memory region stores securely the attestation keys and the timer. It guarantees that device key is accessed only by the PERMANENT protocol resided in ROM. The timer is the component responsible for scheduling RA, thus, it must be tamper-proof and unpredictable by an adversary. To enforce unpredictability, a pseudo-random number generator (PRNG) is used for the time scheduler. Only PERMANENT protocol and blockchain code have read permissions in this memory region.
- **Real Time Clock (RTC).** RTC is a real-time write-protected clock that a software adversary cannot modify. RTC ensures that an attestation response is generated at the current time and the adversary is not reusing old software measurements.

4.2 Security requirements

Considering the adversarial model described in Section 4, we define the required security properties in a blockchain-based RA protocol as follows.

- **Integrity.** The protocol should provide reliable evidence guaranteeing that the attestation response of the *Prv* corresponds to software measurements of the *Prv* at the attestation time (0 when the *Prv* is malicious, 1 otherwise).
- **Integrity of communication data.** The protocol should ensure the *Prv*'s exchanged data cannot be altered without it being detectable by other devices participating in the network.
- **Freshness.** The protocol should ensure that the attestation time is random and confidential. Any given attestation result should be reliably linked to a new attestation time.

5 PERMANENT: Protocol proposal

This section describes in detail the four distinct phases that compose the proposed PERMANENT protocol: (1) Bootstrap, (2) Attestation, (3) Verification, and (4) History-based Trust Decision. Table 3 summarizes the terms used in PERMANENT protocol.

5.1 Bootstrap

The Bootstrap Phase of PERMANENT is an offline procedure executed only once at the beginning of the system deployment. During this phase, the operator OP is responsible for securely deploying the devices, distributing and managing the keys, and installing certificates on the devices. In particular, each device Prv is initialized with an asymmetric signing key pair (SK_{Prv}, PK_{Prv}) and an identity certificate $cert(PK_{Prv})$ signed by OP, guaranteeing that PK_{Prv} belongs to Prv. This certificate is stored in the genesis block, so it cannot be altered and is always available for devices to retrieve. Furthermore, each device is initialized with the Op's public key PK_{OP} to be able to verify $cert(PK_{Prv})$

Table 3. Notation Summary of PERMANENT protocol

Term	Description
Vrf	Verifier
Prv	Prover
OP	System Operator
SK_{Prv}	Secret key of Prover
PK_{Prv}	Public key of Prover
$Block$	The blockchain block containing the attestation data
$PRNG$	Pseudo-Random Number Generator
$Timer$	Scheduled time
$Seed$	The seed of PRNG
$CreatedOn$	Timestamp of the blockchain block
Ψ	Calculated Trust score
α_i	The result of the i'th attestation
n	The number of attestation results stored in the blockchain
$cert(PK_{Prv})$	Identity certificate of Prover
$SeedGenerator()$	A random generator function

of other devices without storing the public key of every participating device. Furthermore, the OP stores a threshold value inside the device to indicate that the device can interact only with other network devices with a trust score above this pre-defined threshold value.

5.2 Attestation

In the following, we describe the attestation of a P2P network with interconnected IoT devices. In such a system, only authenticated device join the permissioned blockchain network and add blocks to the blockchain. However, the blockchain data are publicly readable even from unauthenticated entities. Alternatively, we can consider an IoT system with an edge layer consisting of higher-end edge devices with a larger computational power and storage capacity than the IoT devices. In that case, only a subset of devices deploys the blockchain. To preserve the generality of the approach, in this paper we consider a distributed P2P network where each authenticated device participates in the blockchain.

In PERMANENT, the attestation gets initialized by a timer inside the device. The timer has two functions, scheduling function and triggering function for the attestation. The scheduling function uses a pseudo-random number generator (PRNG) for scheduling the attestation at unpredictable time within a pre-defined time interval. The seed of the PRNG is generated by a random generator function $SeedGenerator()$. We combine the self-attestation procedure and the blockchain to use the same timer.

In PERMANENT, the device performs self-attestation, which means the attestation result is verified by the device itself (secured by a trusted component) and the output of attestation is 0 or 1 (failed or successful attestation). Once the device completes the attestation, it creates a new blockchain block containing the necessary information required to verify the result and device identity. Then, a Merkle Hash Tree is constructed in order to create the header for the block. After the block has been created, it is propagated throughout the network using a gossip protocol. When the block is published, the scheduled time and the seed for the timer is signed and sent along with the block. Based on the PRNG properties, other devices participating in the blockchain use the seed to reproduce the scheduled time and verify that the device was actually allowed to add a new block to the blockchain.

Block design for attestation. The attestation block contains all the data produced during attestation transactions along with the hash of the previous block added during block creation as shown in Figure 2. In particular, the attestation block contains an identifier (i.e., id or public key) for the device that submitted the attestation result, the attestation result, the scheduled time, and the created time. The scheduled time is used during the verification phase to verify the validity of the attestation and detect replay attacks. The created time is used later for the history-based trust decision.

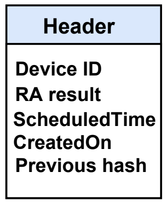

Fig. 2. Data structure for attestation block

5.3 Verification

The verification of published blocks is a three-step procedure: certificate verification, signature verification, and scheduled time verification. In the PoET consensus protocol, a new block is added only when there is the respective devices turn. Thus, the wait time (i.e., the scheduled time) should be verified before adding the block to the blockchain in order to prevent participants from adding a block at arbitrary times. In order to verify the scheduled time, the time and the seed is signed with the private key of the publishing device, using their private key. Since the other devices receive the seed when the block is published, they are able to reproduce and verify the scheduled time. Figure 3 shows the flow of the block verification. The verification procedure starts with a device receiving the scheduled time $Timer$, the seed $Seed$, both signed with Prv's secret key SK_{Prv}, along with the block $Block$. Upon receiving these data, the device first verifies the certificate. If the certificate is valid, then it verifies the signature of

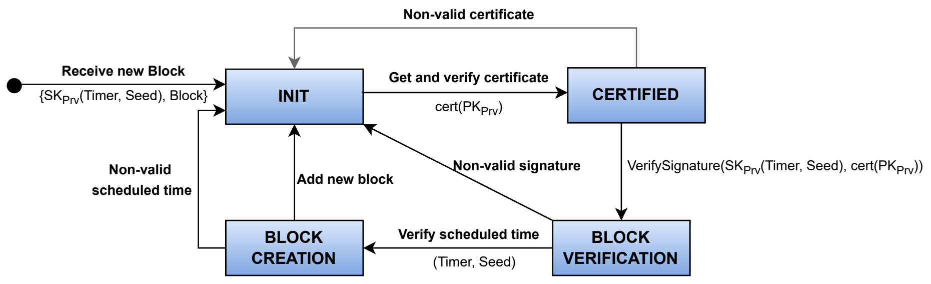

Fig. 3. State Machine - Block Verification

the scheduled time and seed. Once the signature is verified, the device proceeds with the verification of the scheduled time against the creation time *CreatedOn* of the block to ensure the device was allowed to publish a block at this time. If any of the checks fail, then the block is rejected and the device returns the *Init* state, waiting for the next block to be published.

5.4 History-based Trust Decision

The objective of history-based trust decision is to allow devices to decide whether or not to collaborate with another device based on the historical records of the devices attestation stored in the blockchain. The historical records allow the trust decision beyond the recent attestation result. Furthermore, the timestamped blocks in the blockchain allow the attestations to be weighed based on their age, for instance, that older attestations have a lower impact on the trust score.

In PERMANENT, the history-based trust score is a weighted average, where the weight is calculated based on the age of the attestation. In particular, PERMANENT calculates the trust score by taking how long after the genesis block the attestation result was made and divide it by how much time has actually passed since the genesis block was created. Moreover, in PERMANENT, successful attestations have a value of one, while failed attestations have a value of minus one. This means that if a device failed an attestation a long time ago, and after that it has passed successful attestations after that, then the failed attestation should not have the same impact as if the device failed more recently. Note that we assume that after a device has failed the attestation the Network Operator will bootstrap/update the device. In general, the update process is considered out of scope of the RA objective. Thus, we do not provide further process details, but we assume that a recent failed result is a stronger indicator than an old one.

Equation 1 shows the calculation of the trust score.

$$\Phi = \frac{\sum_{i=1}^{n}\left(\frac{CreatedOn_i - CreatedOn_{genesis}}{now - CreatedOn_{genesis}}\right) \times \alpha_i}{n} \mid \alpha_i = \begin{cases} 1 \text{ iff Attestation passed} \\ -1 \text{ iff Attestation failed} \end{cases} \tag{1}$$

where Φ is the resulting trust score, $CreatedOn_i$ is the created on timestamp of the block for attestation i'th, $CreatedOn_{genesis}$ is the created on timestamp of the genesis block, now is the timestamp of the current time, α_i is the result of the i'th attestation and n is the number of attestation results stored in the blockchain. In this equation, α is one if the attestation passed and minus one if it failed.

Figure 4 shows the evolution of the range of trust scores based on the number of attestations evenly distributed over its lifetime. The maximum trust score is shown with green and is the score if all attestations are passed. While all failed attestations are shown in red. Any mix of passed and failed attestation will be within the range between the green and the red graphs. Due to the weight, the

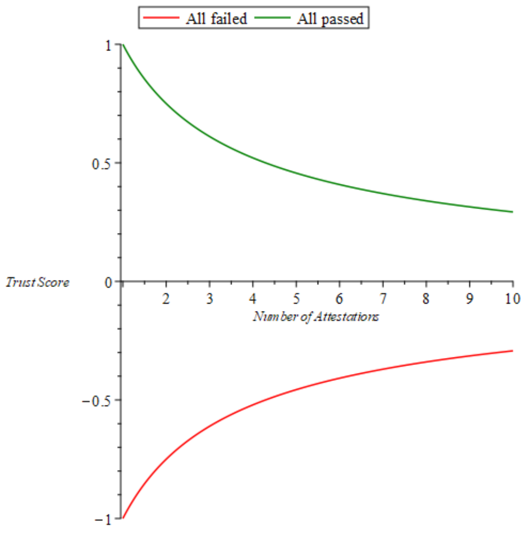

Fig. 4. Trust Score Range

range of the trust score decreases over time. From Figure 4 it is clear that when time passes the devices cannot obtain the same score as when they are newly deployed. This causes some challenges when trying to compare two devices with different ages, even if both devices have passed all attestations.

To mitigate this and make the comparison more clear, the final trust score is divided by the maximum trust score the device in question can obtain, as in Equation 2. In this context, the final trust score shows how close the device is to its maximum trustworthiness. Thus, when two devices passed all their attestations, they will have the same final trust score and will be considered equally trustworthy.

$$\Psi = \frac{\Phi}{\Phi_{max}} \tag{2}$$

where Ψ is the final trust score, Φ is the trust score calculated in Equation 1, and Φ_{max} is the maximum trust score the device can achieve.

The maximum trust score Φ_{max} is calculated as in Equation 1, but with α always equal to one. This means the maximum trust score is determined by the number of attestations and how long ago they where made. The calculation of the maximum trust score is done as in Equation 3.

$$\Phi_{max} = \frac{\sum_{i=1}^{n} \left(\frac{CreatedOn_i - CreatedOn_{genesis}}{now - CreatedOn_{genesis}} \right)}{n} \tag{3}$$

The final trust score can be in the range $[-1, 1]$. It will be negative if the total weighed failed attestations have a higher value then the total weighed passed attestations. This means if a device has a negative final trust score, it is highly untrustworthy. To be trustworthy, a device should be in the positive range, where a threshold for needed trustworthiness can be chosen, e.g, a final trust score of $\Psi \geq 0.5$. Furthermore, the final trust score allows for comparing devices, such that $\Psi_1 = 0.3 > 0.1 = \Psi_2$ means that Device 1 is more trustworthy than Device 2, even though they are both below the threshold.

6 Implementation details and proof of concept

We implemented PERMANENT in Python, using Hyperledger Sawtooth as the underlying blockchain. Hyperledger Sawtooth is a well supported blockchain platform, which can use the PoET consensus algorithm. Docker has been used to deploy each component in separate containers, simulating a network of devices.

The system consists of six components, namely, Validator, Rest-API, Transaction Processor, Settings Processor, and Consensus Engine. Each of these components are deployed in individual Docker containers, while an IoT device can include each component as depicted in Figure 5. Four of the aforementioned components (i.e., Validator, Rest API, Settings Processor, and Consensus Engine) come with the HyperLedger Sawtooth platform and require no changes, while two components (i.e., Client and Transaction Processor) are custom and contain the logic of the application.

6.1 Client

The Client contains the code for interacting with the blockchain. In particular, it is the entity that creates the attestation result and submits it to the blockchain. When the Client starts, it first sets up event subscriptions to listen and receive events from the Validator. After the subscriptions, the Client runs an initial attestation. Then, the Client continuously check if there is a scheduled attestation.

Attestation. The attestation of the Client has three steps: scheduling/ triggering, computing attestation result, and publishing the result in the blockchain. In this implementation, the scheduling is done by using a cryptographically secure pseudo-random generator. Specifically, once an attestation is performed,

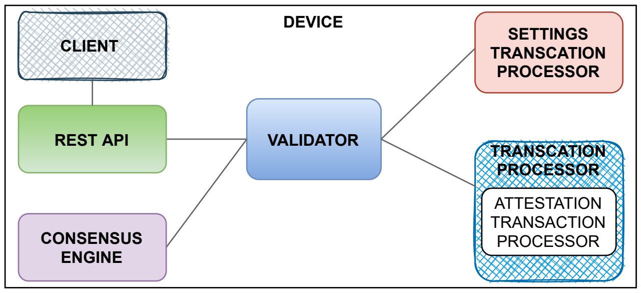

Fig. 5. Hyperledger Sawtooth components for a single device

the schedule is updated by adding a random time to the previously scheduled time. The seed is generated as a random number using the $/dev/urandom$ Linux random number generator. Then, the seed is sent along with the attestation result to allow other devices that know the last scheduled time and the seed to compute and verify the current scheduled time. To compute the attestation, we perform the static software measurement of the device. After the result has been computed, the Client wraps the result, device id, created on date, seed and scheduled time in a transaction with a message identifier, showing it is a *publish attestation* transaction. The Client then wraps the transaction in a batch and sends it to the Rest API.

6.2 Transaction Processor

The transaction processor has two parts: one part handles the business logic for the Attestation transaction family and the other one handles the Diffie-Hellman transaction family. Each of these two parts consists of three components: the *Handler*, *Payload* and *State*. The *Handler* contains the business logic for the transaction family and is the smart contract for the family. The *Payload* defines what the transactions for the family must look like. The *State* contains the getting and setting of the blockchain state data, as well as serializing and deserializing the data.

Attestation. The only business rule for attestations to be accepted to the blockchain is that they have to follow the specified format. This means attestations should specify the attestation message, construct a defined attestation payload and the transaction, and the batch has to be signed by a key accepted in the blockchain network.

7 Limitations

While PERMANENT protocol allows devices to store historical results in the blockchain, the proposed solution also has some limitations.

PERMANENT relies on the PoET consensus algorithm due to the low computational and communications overhead that this algorithm provides. However, PoET has a relatively low throughput compared to other consensus algorithms. This could present an issue if the RA protocol is required to run very frequently in a large-scale network. However, RA protocols typically do not run very often to require high throughput. Thus, this limitation is not critical in the setting where attestations are not performed very often.

Another well-known open research challenge in applying blockchain technology in IoT is the increased memory requirements. Since the blockchain is a distributed ledger, every device needs to store the entire blockchain database with the results of all the devices. If devices have a long lifetime and/or run many attestations, this database could expand rapidly, possibly beyond the available memory of the devices.

Furthermore, the blockchain solution introduces extra cryptographic operations. Cryptographic operations are known to be resource expensive for IoT devices. So the extra security of the blockchain comes at the computational cost of the extra cryptographic operations. However, it may require fewer attestations that also use cryptographic operations, so the total amount of operations might be the same or less over a longer period.

8 Conclusions and Future work

This paper proposes PERMANENT, a decentralized and publicly verifiable remote attestation protocol that relies on blockchain technology. Instead of deciding the trustworthy state of a device based on the latest attestation result, the proposed protocol uses the history of attestations to validate the trustworthiness of each IoT device and calculate the corresponding trust score. We presented the proof-of-concept implementation of the proposed protocol with HyperLedger Sawtooth using Proof-of-Elapsed-Time (PoET) as a consensus mechanism, demonstrating the feasibility of the solution.

While in this paper, we assumed that devices that are trustworthy above a threshold can proceed their interactions, in our future work we will extend the protocol by providing technical details of group session key establishment among trusted devices. Moreover, as future work, we plan to perform some performance optimizations in the proof-of-concept implementation and conduct an empirical analysis of the protocol's performance. Another main area of our future work will be investigating and designing even more efficient blockchain architectures for IoT devices. Furthermore, we will explore and investigate the possibility of attesting devices with lightweight cryptographic operations while providing strong security guarantees.

Acknowledgment

This work is supported by Danish Industry Foundation through project "CIDI - Cybersecure IoT in Danish Industry" (project number 2018-0197) and the European Union's Horizon 2020 Research and Innovation program under Grant Agreement No. 952697 (ASSURED).

References

1. Abera, T., Asokan, N., Davi, L., Koushanfar, F., Paverd, A., Sadeghi, A.R., Tsudik, G.: Invited – Things, Trouble, Trust: on Building Trust in IoT Systems. In: Proc. 53rd Annu. Design Autom. Conf. pp. 1–6 (2016)
2. Abera, T., Bahmani, R., Brasser, F., Ibrahim, A., Sadeghi, A., Schunter, M.: DIAT: Data Integrity Attestation for Resilient Collaboration of Autonomous System. In: Proc. Netw. Distrib. Syst. Secur. Symp. (2019)
3. Ambrosin, M., Conti, M., Lazzeretti, R., Rabbani, M., Ranise, S.: Collective Remote Attestation at the Internet of Things Scale: State-of-the-art and Future Challenges. IEEE Commun. Surv. Tutor. **22**(4), 2447–2461 (2020)
4. Ambrosin, M., Conti, M., Lazzeretti, R., Rabbani, M.M., Ranise, S.: PADS: Practical Attestation for Highly Dynamic Swarm Topologies. In: Proc. Int. Workshop Secure Internet Things (SIoT). pp. 18–27 (2018)
5. Ambrosin, M., Conti, M., Ibrahim, A., Neven, G., Sadeghi, A.R., Schunter, M.: SANA: Secure and Scalable Aggregate Network Attestation. In: Proc. ACM SIGSAC Conf. Comput. Commun. Secur. pp. 731–742 (2016)
6. Ankergård, S.F.J.J., Dushku, E., Dragoni, N.: State-of-the-Art Software-Based Remote Attestation: Opportunities and Open Issues for Internet of Things. Sensors **21**(5) (2021)
7. Arthur, W., Challener, D.: A Practical Guide to TPM 2.0: Using the Trusted Platform Module in the New Age of Security (2015)
8. Asokan, N., Brasser, F., Ibrahim, A., Sadeghi, A.R., Schunter, M., Tsudik, G., Wachsmann, C.: SEDA: Scalable Embedded Device Attestation. In: Proc. 22nd ACM SIGSAC Conf. Comput. Commun. Secur. pp. 964–975 (2015)
9. Bampatsikos, M., Ntantogian, C., Xenakis, C., Tomopoulos, S.C.: BARRETT blockchain regulated remote attestation. Proceedings - 2019 IEEE/WIC/ACM International Conf. on Web Intelligence pp. 256–262 (2019)
10. Brasser, F., El Mahjoub, B., Sadeghi, A.R., Wachsmann, C., Koeberl, P.: TyTAN: tiny trust anchor for tiny devices. In: n Proc. 52nd Annu. Design Autom. Conf. pp. 1–6 (2015)
11. Carpent, X., Rattanavipanon, N., Tsudik, G.: Remote attestation via self-measurement. ACM Trans. Des. Autom. Electron. Syst. **24**(1) (2018)
12. Conti, M., Dushku, E., Mancini, L.V.: Distributed Services Attestation in IoT. In: From Database to Cyber Security, pp. 261–273. Springer (2018), ISBN: 978-3-030-04834-1
13. Conti, M., Dushku, E., Mancini, L.V.: RADIS: Remote Attestation of Distributed IoT Services. In: Proc. 6th Int. Conf. Softw. Defined Syst. (SDS). pp. 25–32 (2019)
14. Dai, H.N., Zheng, Z., Zhang, Y.: Blockchain for Internet of Things: A Survey. IEEE Internet of Things J. **6**(5), 8076–8094 (2019)
15. Dushku, E., Rabbani, M.M., Conti, M., Mancini, L.V., Ranise, S.: SARA: Secure Asynchronous Remote Attestation for IoT Systems. IEEE Trans. Inf. Forensics Security **15**, 3123–3136 (2020)
16. Eldefrawy, K., Tsudik, G., Francillon, A., Perito, D.: SMART: Secure and Minimal Architecture for (Establishing Dynamic) Root of Trust. In: Proc.19th Annu. Netw. Distrib. Syst. Secur. Symp. (NDSS). pp. 1–15 (2012)
17. Favaretto, M., Tran Anh, T., Kavaja, J., De Donno, M., Dragoni, N.: When the Price Is Your Privacy: A Security Analysis of Two Cheap IoT Devices. Advances in Intelligent Systems and Computing **925**, 55–75 (2020)

18. Garcia Lopez, P., Montresor, A., Datta, A.: Please, do not decentralize the internet with (permissionless) blockchains! In: 2019 IEEE ICDCS. pp. 1901–1911 (2019)
19. Giaretta, A., De Donno, M., Dragoni, N.: Adding Salt to Pepper: A Structured Security Assessment over a Humanoid Robot. In: Proc. of ARES 2018 (2018)
20. Halldórsson, R.M., Dushku, E., Dragoni, N.: ARCADIS: Asynchronous Remote Control-Flow Attestation of Distributed IoT Services. IEEE Access **9**, 144880–144894 (2021)
21. Ibrahim, A., Sadeghi, A.R., Tsudik, G.: US-AID: Unattended scalable attestation of IoT devices. In: Proc. IEEE 37th Symp. Reliable Distrib. Syst. (SRDS). pp. 21–30 (2018)
22. Ibrahim, A., Sadeghi, A.R., Zeitouni, S.: SeED: Secure Non-Interactive Attestation for Embedded Devices. In: Proc. of the 10th ACM Conf. on Security and Privacy in Wireless and Mobile Networks WiSec '17. pp. 64–74 (2017)
23. Jenkins, I.R., Smith, S.W.: Distributed IoT Attestation via Blockchain. Proceedings - 20th IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing, CCGRID 2020 pp. 798–801 (2020)
24. Kuang, B., Fu, A., Yu, S., Yang, G., Su, M., Zhang, Y.: ESDRA: An Efficient and Secure Distributed Remote Attestation Scheme for IoT Swarms. IEEE Internet of Things J. **6**(5), 8372–8383 (2019)
25. Neshenko, N., Bou-Harb, E., Crichigno, J., Kaddoum, G., Ghani, N.: Demystifying IoT Security: An Exhaustive Survey on IoT Vulnerabilities and a First Empirical Look on Internet-Scale IoT Exploitations. IEEE Commun. Surv. Tutor. **21**(3), 2702–2733 (2019)
26. Park, J., Kim, K.: TM-Coin : Trustworthy Management of TCB Measurements in IoT. In: 2017 IEEE PerCom Workshops. pp. 654–659. IEEE (2017)
27. Rabbani, M.M., Dushku, E., Vliegen, J., Braeken, A., Dragoni, N., Mentens, N.: RESERVE: Remote Attestation of Intermittent IoT Devices. In: Proc.19th ACM Conf. Embed. Networked Sens. Syst. (SenSys). pp. 578–580 (2021)
28. Rasolroveicy, M., Fokaefs, M.: Performance evaluation of distributed ledger technologies for IoT data registry: A comparative study. Proc. of WorldS4 2020 pp. 137–144 (2020)
29. Seshadri, A., Perrig, A., Doorn, L.v., Khosla, P.: SWATT: softWare-based attestation for embedded devices. In: IEEE S & P 2004. pp. 272–282 (2004)
30. Sokolov, S., Gaskarov, V., Knysh, T., Sagitova, A.: IoT Security: Threats, Risks, Attacks. Lecture Notes in Civil Engineering **130 LNCE**, 47–56 (2021)
31. Østergaard, J.H., Dushku, E., Dragoni, N.: ERAMO: Effective Remote Attestation through Memory Offloading. In: IEEE International Conference on Cyber Security and Resilience (IEEE-CSR). pp. 73–80 (2021)