Article | Received 13 December 2024; Accepted 11 April 2025; Published 30 April 2025
https://doi.org/10.55092/blockchain20250009

Trust@TEE.TIME: a platform for embedded elapsed time schemes in trusted execution environments for blockchain †

Quentin Jayet[1,*], Christine Hennebert[1], Yann Kieffer[2] and Vincent Beroulle[2]

[1] Univ. Grenoble Alpes, CEA, Leti, F-38000 Grenoble, France

[2] Univ. Grenoble Alpes, Grenoble INP, LCIS, 26000 Valence, France

† This manuscript is an extended version of the paper presented at the 2024 IEEE International Conference on Blockchain (Blockchain) and has been granted copyright permission for publication.

* Correspondence author; E-mail: quentin.jayet@cea.fr.

Highlights:

- Design of a novel verifiable elapsed time proof for blockchain consensus among resource-constrained peers.
- Description of an experimental platform for integrating proof mechanisms based on secure hardware components.
- Analysis of energy consumption and time distribution of several proof mechanisms.

Abstract: consensus protocols and peer-provided proofs are the necessary components used to establish trust in blockchain systems. Verifiable proofs that guarantee an elapsed time based on security components have emerged as alternatives to energy-intensive Proof of Work (PoW). These proofs are well-suited for embedding consensus protocols running on resource-constrained devices. This paper proposes Trust@TEE.TIME, an experimental platform designed to instrument and characterize several proof mechanisms within embedded devices. Our platform is proof-agnostic, allowing it to accommodate various proof generation mechanisms without modifications to the underlying hardware or architecture. It comprises Systems on Module featuring an ARM Cortex-A7 processor with a Trusted Execution Environment (TEE) and a Trusted Platform Module (TPM). These components are collaboratively used to ensure secure proof generation. This paper is an extension of a previous paper which introduced Proof of Hardware Time (PoHT). It provides a more detailed description of the experimental platform and a comparison of power consumption and time delay with different average elapsed time between blocks. In addition, we show that PoHT achieves an average power reduction factor of 7 to 117 compared to PoW.

Keywords: blockchain; embedded systems; elapsed time; trusted execution environment; energy consumption; trusted platform module; proof of hardware time

1. Introduction

Distributed ledger technologies, such as blockchains, employ consensus mechanisms to create an immutable register of transactions that are ordered, timestamped, and authenticated. Transactions are aggregated into blocks, which are issued at regular intervals. Each block includes the hash of the latest block, thereby creating an immutable link between successive blocks. In order to choose which block is accepted, blockchain employs a consensus mechanism, which comprises two distinct components: the consensus protocol, which determines if a block is valid, and the proof mechanism, which guarantees that a certain amount of time has elapsed since the last block. This guarantee of elapsed time is critical for the security of the blockchain. Without it, an attacker could potentially create an arbitrary number of blocks and generate a longer chain than the valid one, thereby bypassing the previously accepted blockchain. This type of attack is known as a long range attack. Proof of Work (PoW) not only ensures consensus through the protocol of Nakamoto [1] but also secures the blockchain against these vulnerabilities by guaranteeing by design the elapsed time between blocks. This elapsed time is guaranteed thanks to a proof value obtained through a large amount of non-optimisable computations to find a solution to a cryptographic puzzle [1]. This approach ensures the elapsed time, but consumes a significant amount of energy. Finding an alternative to PoW that provides at least an equivalent level of security but with low energy consumption would open the possibility of embedding a blockchain protocol in constrained devices. This can bring trust and traceability into networks of embedded physical devices, for instance, in vehicular applications.

Embedded systems, ranging from autonomous devices to edge or IoT devices, are often resource-constrained, with limited processing power, memory, and energy. On the other end, the network of peers forming a blockchain is typically perceived as a solution relying on powerful servers and robust infrastructure. Despite this, blockchain technology could offer significant benefits to the embedded world, particularly for use cases involving the sharing of data or evidence, e.g., enabling tamper-proof logs in autonomous vehicles [2]. These use cases are typically addressed by integrating a lightweight blockchain client into embedded systems [3], enabling the creation of decentralised networks. Going a step further by embedding blockchain directly into the devices themselves could reduce the need for heavy infrastructure and empowering devices to form a distributed network of lightweight peers. In this context, traditional consensus mechanisms like PoW are not suitable, as they require significant computational resources and energy to ensure the security of the blockchain. Therefore, more energy efficient alternatives must be explored to bring blockchain technology to these low-power environments, while maintaining the security and integrity of the network. To reduce power consumption, alternatives to PoW based on security components, such as Trusted Execution Environment (TEE) have emerged [4]. A TEE, such as ARM TrustZone, is a secure execution environment within a CPU. It is commonly found in devices with applications requiring a high level of security, such as biometric authentication or secure IoT. The TEE operates alongside the Rich Execution Environment (REE) of the device. The TEE and the REE are isolated within the CPU, each environment having dedicated memory, and execution contexts, preventing unauthorised interaction. The role of the TEE is to store, process, and protect sensitive code in a trusted environment. Trusted applications within the TEE benefit from a range of security services such as code execution integrity, secure data storage, and management of cryptographic keys and algorithms.

The REE is an unsecured location where the Operating System (OS) and non-secure applications run. Other kinds of trusted hardware are available, such as the Trusted Platform Module (TPM), which is a standardised secure cryptoprocessor defined by the Trusted Computing Group (TCG) [5]. The TPM is designed to provide hardware-based security for key storage and platform integrity. Moreover, secure hardware provides attestation mechanisms that allows them to cryptographically prove their integrity and authenticity to remote parties, ensuring trust and protecting against unauthorised modifications [6].

Our objective is to develop a lightweight blockchain tailored for resource-constrained devices. To achieve this, we aim to construct a verifiable proof of elapsed time based on secure hardware components. The absence of standardized experimental platforms makes it difficult to evaluate and compare different approaches. This paper addresses the challenges of instrumenting, characterising, and implementing proof generation mechanisms within Trusted Execution Environments (TEEs). In particular, it introduces Trust@TEE.TIME, an experimental platform designed to facilitate the integration of proof mechanisms into embedded secure hardware components. Instead of relying exclusively on TEEs, Trust@TEE.TIME incorporates a Trusted Platform Module (TPM) to provide standardised attestation capabilities. Trust@TEE.TIME has been designed to be proof-agnostic, offering flexibility in the proof generation process. This process can leverage various software and hardware functionalities, including signature generation, hashing, and communication with the TPM.

This paper extends our previous work, which introduced Proof of Hardware Time (PoHT) as a proof mechanism based on elapsed time, specifically designed for embedded systems. Additionally, the previous work compared various proof mechanisms to evaluate their suitability for such systems. This paper extends our paper by providing a complete description of our experimental platform Trust@TEE.TIME. To this end, this paper presents the following contributions: (1) the experimental platform Trust@TEE.TIME, (2) a refined model of the elapsed time for PoW, Proof of Sequential Work (PoSW) and PoHT and (3) a comparison of the energy consumption and the reliability of the elapsed time measurements for the three proofs for different average elapsed times between blocks.

This paper is structured as follows: Section 2 introduces the importance of the elapsed time in blockchain and various methods to attest and guarantee it. Section 3 describes Trust@TEE.TIME, an experimental platform for instrumenting and characterising proof generation in embedded devices. Section 4 explores elapsed time modelling for three distinct elapsed time based proofs. Section 5 presents the comparison of the power consumption and the experimental distributions of elapsed time between consecutive blocks. Finally, Section 6 concludes and discusses perspectives.

2. Related work

This section outlines the importance of the elapsed time in blockchains and provides a review of some consensus mechanisms used in embedded systems.

2.1. Elapsed time between blocks in blockchain

2.1.1. Time as a security asset

The finality of a blockchain is defined as the point in time when a block becomes impossible to remove from the blockchain [7]. Blockchain can be divided into two principal categories: those with immediate

finality and those with probabilistic finality. In systems with immediate finality, once a block is added to the blockchain, it becomes permanent and cannot be revoked. This is achieved through periodic consensus among all peers, who vote on the next block to append. However, if consensus is not reached during a given round, a new leader is selected, and another attempt is made to achieve consensus. During this process, the system's availability is temporarily interrupted. While the elapsed time between block elections is typically constant, the system's availability is not always guaranteed, which can result in longer intervals. In systems with probabilistic finality, a divergence in the blockchain's structure can occur. This is called a fork and is caused when the chain splits into two separate paths. In accordance with the longest chain rule, peers converge on the longest chain of blocks, resulting in the revocation or replacement of blocks on shorter chains that originate from forks. Consequently, blocks may be revoked or replaced after some time. However, the longer a block remains on the chain, the more challenging it becomes to revoke. In such systems, the time between blocks is not fixed but probabilistic, with only the average time interval being known.

A long-range attack occurs when an attacker creates a fork starting from an earlier block in the blockchain and succeeds in generating a branch longer than the legitimate one [8]. The time between two blocks is an important security asset for blockchain, particularly against such attacks. Guaranteeing the elapsed time between consecutive blocks ensures that the actual time between blocks cannot be made shorter, preventing an attacker from creating a branch longer than the honest one.

2.1.2. Guaranteeing the elapsed time

In Proof of Work (PoW), each block contains a verifiable proof value consisting of a nonce, *i.e.*, a value appended in the block header. This nonce represents a solution of a cryptographic puzzle that satisfies common rules, verifiable by everyone through simple verification technique [1]. The time required to solve the puzzle regulates the time elapsed between two consecutive blocks. A parameter, called the difficulty, adjusts the amount of computation required to statistically find a solution in the allotted time, using all the computational power of the peer nodes participating in the protocol. Thus, the nodes in the network participate in a lottery to add their block, where the winner is determined the first node to find a valid solution to the cryptographic puzzle. While this principle ensures a probabilistic minimum elapsed time between two blocks, it allows delegation to specialised hardware and parallelisation of the nonce drawing, leading to the construction of mining farms and to an excessive energy consumption.

Time-lock puzzles aim to guarantee an elapsed time through sequential computations. They were originally used in timed-release cryptography to cipher data that can only be deciphered after a certain amount of sequential computation [9]. However, the elapsed time in a time-lock puzzle cannot be verified by a third party. Boneh *et al.* [10] formalises Verifiable Delay Functions (VDFs). VDFs are sequential functions analogous to time-lock puzzles. They provide a verifiable proof of the sequence of computations performed. The purpose of VDFs is to guarantee a series of sequential computations, not to design a lottery winner. Therefore, to include a VDF in a consensus protocol, VDF requires the generation of a random number as in [11], where a VDF is used to create a Proof of Behavior model [12]. To this end, Proof of Sequential Work (PoSW) aims to create a sequential proof for a consensus protocol [13]. While these methods are non-parallelisable by design, they still require energy-intensive computations to be performed.

In contrast, proofs with immediate finality are more sensitive to long-range attacks, as they do not guarantee the elapsed time by design [14]. Proof of Stake (PoS) relies heavily on software to guarantee the finality [15], thereby significantly increasing the attack surface [16].

Guaranteeing the elapsed time is crucial for preventing long-range attacks and securing the blockchain. However, existing solutions require a substantial amount of energy to assure by design the elapsed time. These solutions can not be used in the context of resource constrained devices.

2.2. Consensus mechanisms for embedded systems

Some research efforts focus on the design of consensus mechanisms specifically adapted to embedded devices. Among these, [17], [18], and [19] propose approaches based on Proof of Work (PoW) and Proof of Sequential Work (PoSW). However, these methods suffer from high energy consumption, which negatively impacts the battery life of on-board devices. Dorri *et al.* [20] propose a distributed time-based consensus which reduces the mining processing overhead but may be vulnerable to Sybil attacks in small networks. Bada *et al.* [21] present a framework for selecting an appropriate blockchain, taking into account the specific use case and the associated energy consumption, highlighting Proof of Authority (PoA), Byzantine Fault Tolerance based blockchains and Proof of Elapsed Time (PoET). In [22], the authors concentrate on the energy consumption of various PoS blockchains, but the comparison is based on theoretical modelling and not on experimental measurement on a running blockchain. A comparison of a variety of consensus protocols in order to create a fully autonomous IoT network is presented in [23]. The comparison made is mainly based on the scalability, computational overhead and network communication. The authors identify three consensus protocols that are suitable for such a scenario: Practical Byzantine Fault Tolerance (PBFT), Tangle, and Proof of Elapsed Time (PoET). The remainder of this section discusses PBFT, Tangle, incentive-based proofs, and hardware-based proofs such as PoET in the context of creating a verifiable proof for embedded systems.

2.2.1. Practical byzantine fault tolerance

Castro *et al.* [24] introduce Practical Byzantine Fault Tolerance (PBFT), which is a consensus method designed to ensure the security and reliability of distributed systems in the presence of Byzantine faults. In this multi-round process, all nodes need to communicate and vote on the proposed transaction during an appointment, involving synchronisation of devices. PBFT requires a lot of peer-to-peer communications, which increases the energy consumption and makes it unsuitable for embedded systems.

2.2.2. Tangle protocol in Internet of Things Application

The Tangle protocol serves as the consensus mechanism for the Internet of Things Application (IOTA) [25], distinguishing itself from traditional blockchain systems. In contrast to block-based architectures, which organise transactions into discrete blocks, Tangle employs a directed acyclic graph (DAG) structure. In this system, each new transaction must validate two prior transactions, effectively linking them within the graph and ensuring the decentralised nature of the consensus process. This approach enhances scalability and efficiency, as transactions are not limited by block formation but can continuously build upon one another in the graph. However, despite its DAG-based structure, Tangle requires the presence of a single

coordinator [26]. Controlled by the IOTA Foundation, the coordinator issues trusted milestones that confirm and establish consensus, securing the network and preventing attacks. While this coordinator facilitates the stability of the network, its centralised control over the consensus compromises the decentralisation of the network by introducing a single point of authority. Consequently, the Tangle protocol's reliance on a coordinator recentralises the system, which deviates from the fully decentralised principles typically associated with blockchain technologies.

2.2.3. Incentive-based proofs

IOTA is moving forward with IOTA 2.0, which integrated the Delegated Proof of Stake (DPoS) mechanism into its framework. DPoS is a derivative of Proof of Stake (PoS) [15], a consensus algorithm in which participants validate blocks based on the amount of currency they are holding, called stakes, rather than their computational power. DPoS refines this concept further by introducing a delegated layer, whereby a limited number of trusted nodes are elected by the stakeholders to validate blocks on behalf of the network. The ledger Hedera utilises a hashgraph to reach a high transaction throughput, and operates on the basis of DPoS for achieving consensus [27].

Although these approaches eliminate the computational demands associated with PoW, they are not designed to guarantee the elapsed time between blocks. Consequently, additional mechanisms are required to guarantee the finality. In particular, to prevent peers from cheating and producing blocks at a higher rate than usual, participants attempting to defraud the system are penalised by having their stakes confiscated. So, in incentive-based proofs, long-range attacks are not countered by designing a proof that guarantee the elapsed time but by creating software countermeasures in the consensus protocol that demotivate peers to cheat. To prevent long-range attacks at the proof level, the proof mechanism must be designed to enforce a verifiable elapsed time, ensuring that an attacker cannot shorten the interval between successive blocks. This preserves the integrity and security of the historical record of events.

2.2.4. Hardware-based proofs

The idea of leveraging hardware security components to guarantee a time interval, rather than relying on extensive cryptographic computations, emerged in 2016 with Milutinovic's Proof of Luck (PoL) [4]. PoL utilizes a Trusted Execution Environment (TEE) to generate a random number, called luck, with the rule that the node with the highest value will add the next block to the chain. The validity of the blockchain depends on the cumulative luck of a chain. However, PoL remains a prototype [28] with a simulated TEE and has not been deployed in a production blockchain.

Intel later introduced Proof of Elapsed Time (PoET) in Hyperledger Sawtooth [29], using Intel SGX to generate a random wait time within a secure enclave. The first node to wake up and provide an attestation from its SGX enclave earns the right to append a block. However, PoET is manufacturer-dependent and does not scale well. In 2017, Chen identified vulnerabilities when a small fraction of nodes is compromised, though Bowman *et al.* [30] later provided a formal proof of a resilient elapsed-time consensus protocol.

According to NIST, an attestation involves generating a digital signature for securely stored hardware measurements, allowing the verification of the authenticity by a requester [31]. To achieve

a manufacturer-agnostic scheme, trust in the TEE must be verified via remote attestation [6]. Such attestations exist for Intel SGX (EPID) [32], ARM TrustZone [33], and TPMs [5]. Ankergård [34] combined self-remote attestation with PoET's waiting time to create a blockchain certifying IoT device trustworthiness. However, it lacks the ability to dynamically add devices, requiring a new blockchain for each IoT deployment.

Proof of Hardware Time (PoHT) is a proof based on the measurement of an elapsed time within an embedded device with a TEE and a TPM [35]. The reliability of elapsed time measurement in PoHT has been analysed in [36]. Under a temperature attack, clock drifts occur, making the measurement unreliable. Without network synchronization, a single clock cannot detect this drift. However, leveraging multiple clocks within a device enables drift detection when the device is under attack.

2.3. Comparison of state-of-the-art consensus methods

Table 1 illustrates the different characteristics of various consensus methods. PoHT is characterised by low computation, manufacturer independence, and non-parallelisability. In contrast, PBFT is low computation and non-parallelisable but high communication. IOTA and PoS offer low computation and low communication, but are software implemented, leading to a larger attack surface. PoET provides low computation and low communication, but relies on Intel SGX, making it manufacturer dependent.

Table 1. Comparison of state-of-the-art consensus methods.

Consensus Method	Computation	Communication	Secure Components	Manufacturer Dependency	Parallelism	Mechanism
PoW	High	Low	No	No	Yes	Lottery
PoSW	High	Low	No	No	No	Lottery
PBFT	Low	High	No	No	No	Leader Election
PoS	Low	Low	No	No	Yes	Leader Election
DPoS	Low	Low	No	No	Yes	Leader Election
IOTA	Low	Low	No	No	Yes	Coordinator
PoL	Low	Low	Yes	Yes	-	Lottery
PoET	Low	Low	Yes	Yes	-	Lottery
PoHT	Low	Low	Yes	No	No	Lottery

We aim to compare probabilistic proofs designed to guarantee time by design. Our focus is on Proof of Work (PoW), which is parallelisable and energy intensive, Proof of Sequential Work (PoSW) which is non-parallelisable and energy intensive, and Proof of Hardware Time (PoHT), which is non-parallelisable and low power, making it suitable for embedded systems.

3. Trust@TEE.TIME platform

This section outlines the experimental platform, which facilitates the experiments of several embedded proofs. In order to ensure the reliability of the results, the proofs mechanisms are subjected to the same experimental conditions, enabling a comparison where only the selected characteristics impact the results.

3.1. Description of the Trust@TEE.TIME platform

3.1.1. Proof mechanism in blockchain architecture

Our end goal is to develop a lightweight blockchain suitable for embedded systems. In our study, we focus on the proof generation within an embedded device, specifically on the characteristics that guarantee the elapsed time between two blocks and its power consumption. The consensus protocol to designate the winner (*i.e.*, the peer which can add its block to the blockchain) is not considered at this stage.

A blockchain is structured in layers as shown in Figure 1. Each layer of the blockchain performs a specific function. The bottom layer is the physical layer, which comprises a peer-to-peer (P2P) network that enables the transmission and reception of blocks and data between nodes. Above this layer is the validation layer, which supports the consensus mechanism of the blockchain. Consensus mechanisms are composed of a local process (peer embedding process) for generating proof and a consensus protocol between distributed peers. The proof generation process generates a proof value that guarantees that enough time has passed since the last block. The consensus protocol uses the generated blocks and their associated proof values to determine which block is the most legitimate to be added to the blockchain, according to common rules. This protocol ensures that the winning block is designated by the majority of peers. The service layer consists of smart contracts that are deployed and accessible on the blockchain, providing functionality that is directly available to users. These smart contracts facilitate interactions with the blockchain by allowing code execution and data storage directly within the blockchain. Finally, the user layer includes all user devices that can communicate with the service layer through transactions.

The proof generation mechanism is supported by each peer node participating in the validation. An experimental platform is developed to facilitate the design of a low power proof for embedded systems, enabling easy instrumentation, firmware updates, and trusted application updates for testing and characterization of embedded proofs. In particular, this entails monitoring the power consumption and elapsed time characteristics of the considered proofs.

3.1.2. Global architecture of Trust@TEE.TIME

The objective of this study is to propose a method to experiment and characterise the proof generation process within embedded device. To achieve this, we design a solution that involves performing only the proof generation process in the embedded device. This approach enhances the precision of power consumption measurements by limiting the activity of validators to the proof generation only. In order to facilitate the evolution of this platform towards the structure of a blockchain with a consensus protocol, smart contracts from the service layer are employed to collect proofs and headers in the absence of a consensus protocol. Once the consensus protocol has been described, the smart contracts will be incorporated into the validator layer, and the proof can be implemented in real blockchain layers.

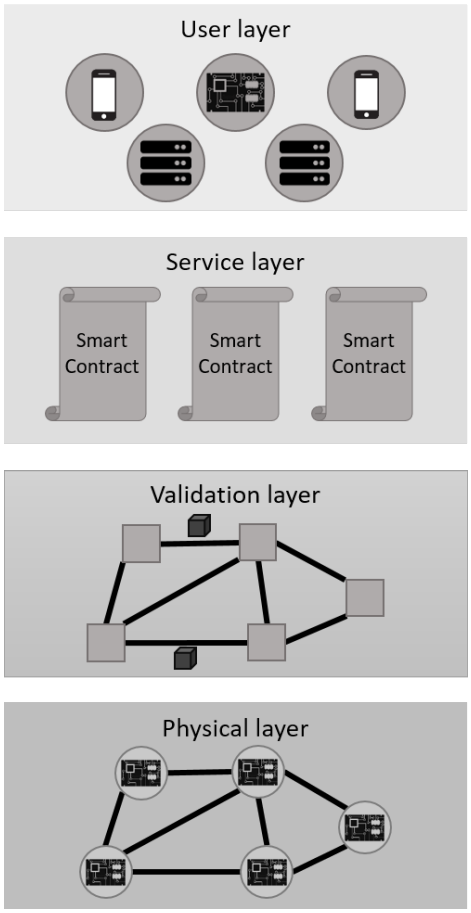


Figure 1. Blockchain layers: users in the user layer send transactions to the service layer composed of smart contracts, validators in the validation layer are responsible for executing the consensus mechanism, the underlying physical layer allows peers to communicate in a P2P network.

In Trust@TEE.TIME, the network is structured as a distributed P2P system composed of validators and a computer. The validators are embedded devices equipped with security components, namely an ARM TrustZone and a TPM. The TrustZone is employed to securely execute the proof generation process, while the TPM provides standardised attestation mechanisms. The computer operates the Ethereum simulator Ganache [37], which is a platform hosting the simulated ledger and executing smart contracts as part of the physical layer.

Figure 2 presents the architecture of the Trust@TEE.TIME platform. Users (user layer) send transactions to a smart contract (service layer). The transactions are transmitted to the peer validators. Each validator is an embedded device that builds its own *candidate_block* from the transactions received, *i.e.*, a block without a proof value. Once a *candidate_block* is built, the process of proof generation starts involving the TEE of the concerned peer. After the proof value has been generated, it is included in the header of the *candidate_block*, forming a complete block ready to be submitted to the consensus of peers. As the aim of our platform is to focus on the proof and the elapsed time between the successive blocks, only the header of each block is exchanged between peers. To this end, a smart contract is used to record the header of the submitted blocks in the ledger.

Local measurements are conducted on the embedded peer without interference from other programs,

thereby ensuring that the results are not influenced by external processes. Furthermore, global measurements are performed at the simulation layer, thus allowing access to global information such as the elapsed time between blocks. In conclusion, measurements are conducted exclusively on the proof generation mechanisms within an embedded peer, thereby facilitating a more comprehensive understanding and characterisation of the proof generation process.

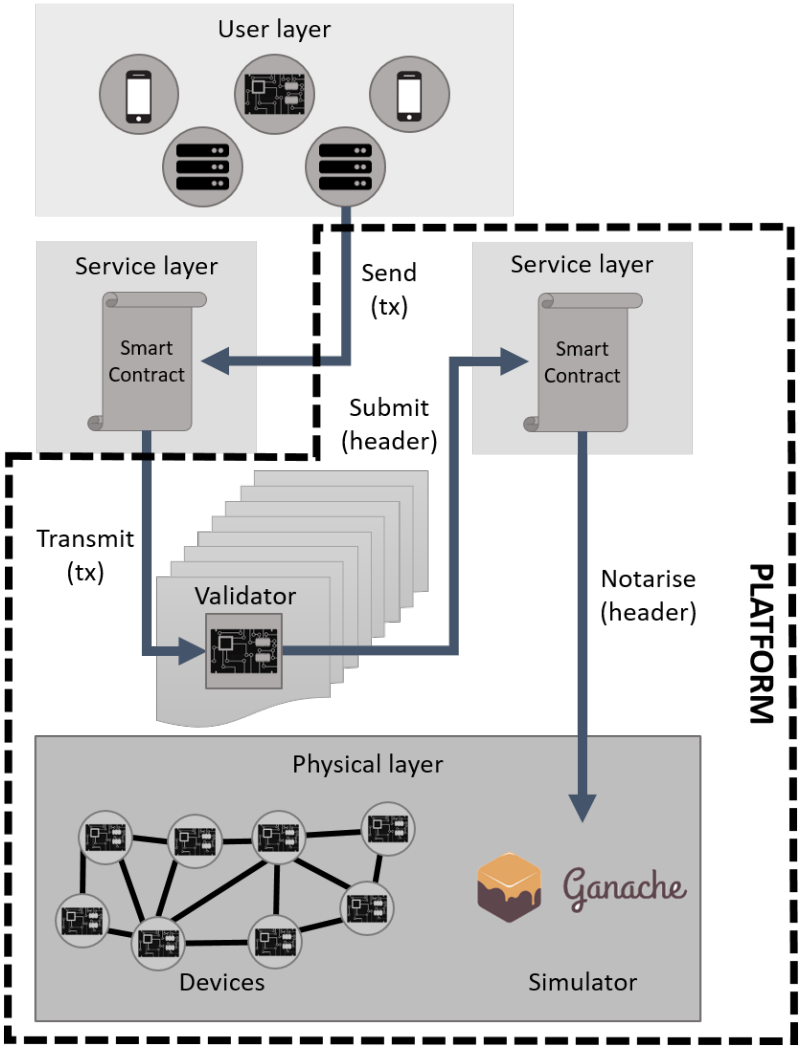


Figure 2. Architecture of the experimental platform Trust@TEE.TIME.

3.2. Embedded architecture of Trust@TEE.TIME

In Trust@TEE.TIME, each validator node is a STM32MP157f-dk2 electronic board, standing as a System-on-Module (SoM), composed of an ARM Cortex-A7 microprocessor and a daughter board integrating a Trusted Platform Module (TPM). The microprocessor consists of a Normal World embedding a Linux Operating System (OS), which is considered unsecured, and a Secure World, a TrustZone running an OP-TEE OS [38]. The TrustZone is a security mechanism designed to create a secure execution environment within ARM processors. This environment is isolated from the rest of the system through a software-enforced privilege check, ensuring that only authorised code can access the Secure World. This isolation enables the secure execution of critical functions, such as the management of secret data for encryption or digital signature operations, the processing of sensitive information,

and the running of critical secure applications. The TPM is a standardised hardware component, providing a secure cryptoprocessor for applications such as key management and attestations. In addition, the TPM can be accessed only from the TrustZone of the microprocessor through a secure Serial Peripheral Interface (SPI) bus.

The embedded architecture, depicted in Figure 3, is designed to be proof-agnostic, meaning that the proof generation component is modular and interfaces seamlessly with the rest of the system. Blocks are received in the Normal World by an event from the smart contract layer, triggering the proof generation process in the TrustZone. This process has access to a variety of functions, including those available through the TrustZone and those provided by the TPM such as ECDSA signature to send transactions to the Ethereum simulator, communication with the TPM, random number generation and access to hardware timers. When the proof is complete, the proof is incorporated into the block. The block is sent to the smart contract through transaction built and sent via the Ethereum JSON RPC communication stack integrated in the Normal World.

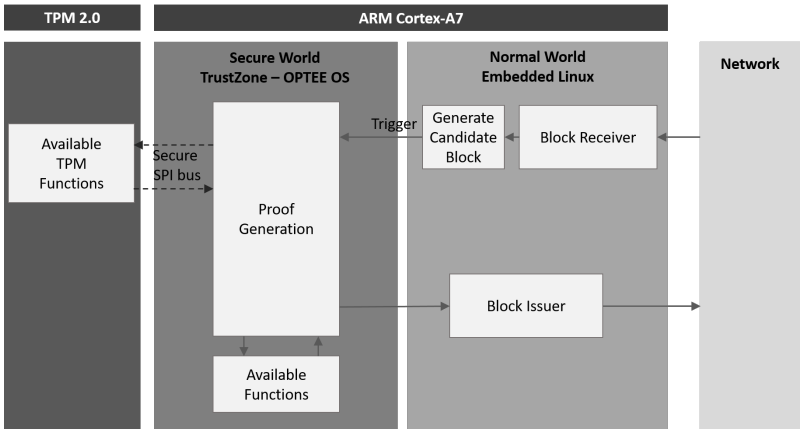


Figure 3. Proof-agnostic embedded architecture of a device validator composed of a System on Module with the Normal World, the Secure World, the TPM and the different functions.

In conclusion, Trust@TEE.TIME functions as a local mechanism enabling proof generation within a TrustZone, as well as a global architectural framework for characterising the proof generation process in an embedded peer. To ensure comprehensive evaluation, measurements are conducted at both the global and local levels of the platform. The current implementation consists of eight nodes, but its design allows for easy scalability by simply adding more nodes. While our implementation is specific to our System on Module (SoM), the architecture is inherently portable to other boards equipped with a TrustZone and a TPM, ensuring broader hardware compatibility.

4. Presentation of the proofs

The experimental platform can be used to assess probabilistic time-based proofs, thereby focusing on probabilistic proofs. In this section, three probabilistic proof generation mechanisms are considered: a Proof of Work (PoW) that is parallelisable and energy-consuming, a Proof of Sequential Work (PoSW) that solves the issue of parallelisation, and the Proof of Hardware Time (PoHT) that is not parallelisable and low power. These three mechanisms are described, their elapsed time modelled and embedded in the

experimental platform.

4.1. Theoretical description of the proofs

4.1.1. Proof of work

Algorithm 1 presents the PoW algorithm considered. PoW relies on the parallelisable computation of a hash function to find a solution to a cryptographic challenge. It requires the generation of a candidate block consisting of: the *version* of the algorithm, the hash of the previous block *previous_hash*, the Merkle root hash *merkle_root* of the transactions Merkle Tree, the *time* and the *difficulty*. The proof value consists of a *nonce* that is randomly found so that the *SHA-256* hash of the complete block is smaller than the *difficulty*. The time needed to find a suitable *nonce* value guarantees the elapsed time between two consecutive blocks and, therefore, makes it very difficult to compress time and perform a long-range attack. This time can be adjusted by changing the *difficulty*.

Algorithm 1 Proof of work algorithm

Require: *candidate_block*
1: $nonce \leftarrow 0$
2: $hash \leftarrow SHA\text{-}256(candidate\_block \,||\, nonce)$
3: **while** $hash \leq difficulty$ **do**
4: $\quad nonce \leftarrow nonce + 1$
5: $\quad hash \leftarrow SHA\text{-}256(candidate\_block \,||\, nonce)$
6: **end while**
7: sign *block*

4.1.2. Proof of sequential work

A PoSW based on a simple Verifiable Delay Function (VDF) is presented in Algorithm 2. The VDF provides the verifiable proof that a sequential computation has been performed. To compare a sequential approach with PoW, the VDF must be probabilistic, thus our version performs a random draw to determine the number of sequential iterations the PoSW must perform. PoSW requires the generation of a candidate block composed of: the *version* of the algorithm, the hash of the previous block *previous_hash*, the Merkle root hash *merkle_root* and the *time*. The proof value consists of the *nonce* and a 256-bits number *proof* resulting from *nonce* sequential hashing operations. This implementation of the PoSW enables the verification of the proof value by a peer node through the re-computing of the hash sequence once the block has been received. Other PoSW based on more complex VDF can be implemented, such as in [39] and [40], to facilitate quicker verification of the proof correctness.

Algorithm 2 Proof of Sequential Work Algorithm

Require: *candidate_block*
1: $nonce \leftarrow$ random from $\mathscr{P}_{PoSW}$
2: $intermediate \leftarrow candidate\_block \,||\, nonce$
3: $i = 0$
4: **while** $i \leq nonce$ **do**
5: $\quad intermediate \leftarrow SHA\text{-}256(intermediate)$
6: $\quad i = i + 1$
7: **end while**
8: $block \leftarrow candidate\_block \,||\, nonce \,||\, intermediate$
9: sign *block*

4.1.3. Proof of hardware time

PoHT is described in Algorithm 3 and consists of generating a random waiting time *wait_time*, waiting until *wait_time* has passed. The time is provided from TPM 2.0 time attestation services [41]. The header of the PoHT is the concatenation of the hash of the previous block *previous_hash*, the Merkle root hash *merkle_root*, the random *wait_time* and the *attestations*. The proof value consists of the *wait_time* and the *attestations*. The *attestations* are composed of two attestations provided by the TPM through a quoting command : (1) the time attestation *start_time* for the beginning of the waiting period and, (2) the time attestation *end_time* at the end of the waiting period. The quoting command is used through the endorsement hierarchy of the TPM, meaning that only the TPM could have signed the time attestations with the Attestation Key (AK). The creation and verification of AK is described by the Trusted Computing Group (TCG) [41]. So the quoting command issues an attestation including the time of the TPM clock and the associated monotonic counter. This time is relative to the TPM environment and cannot be reset without incrementing the monotonic counter. For a given value of the counter, the difference between the *end_time* and the *start_time* provides the elapsed time attested by the secure hardware component. In Algorithm 3, *header_proofless* represents the header without the *wait_time* and the *attestations*.

Algorithm 3 Proof Of Hardware Time Algorithm

Require: *candidate_block*
1: $wait_time \leftarrow$ random from $\mathscr{P}_{PoHT}$
2: $start\_time \leftarrow TPM\_GetTime$
3: $wait(wait_time)$
4: $end\_time \leftarrow TPM\_GetTime$
5: $attestations \leftarrow start\_time \,||\, end\_time$
6: $block \leftarrow candidate\_block \,||\, wait\_time \,||\, attestations$
7: sign *block*

4.2. Elapsed time modelling

4.2.1. Proof of work modelling

To model the PoW described in algorithm 1, let *SHA-256*: $\{0,1\}^* \rightarrow \{0,1\}^{256}$ be the hashing function considered in the random Oracle model. The output follows the uniform probability distribution denoted $\mathscr{U}([0,2^{256}-1])$. Let $\mathscr{X} \sim \mathscr{U}([0,2^{256}-1])$ be the random variable representing the block hash. For $n \in [0,256]$, the difficulty d_n of the PoW is a value such that if the block value is less than d_n, then the block written in binary starts with n zeros, so, $d_n = 2^{256-n}-1$.

$$p = P(\mathscr{X} \leq d_n) = \frac{d_n+1}{2^{256}} = \frac{1}{2^n} \tag{1}$$

Finding a solution to the PoW with one iteration follows a Bernoulli trial of parameter p. Equation 1 provides the value of p as a function of n. Considering that Bernoulli trials are independent, the probability of the first success follows the geometric distribution $\mathscr{G}(p)$. The average number of iterations to win, and the difficulty, are linked by:

$$E(\mathscr{G}(p)) = \frac{1}{p} = 2^n \tag{2}$$

Knowing the time it takes to perform a hash iteration, it is possible to modify the difficulty n in order to increase or decrease the average elapsed time between successive blocks. Equation 2 can be used to adjust the elapsed time between consecutive blocks in the PoW.

Given that the value of n can only take 257 distinct values, the potential average elapsed times are constrained to a discrete set of values. This leads to a restricted range of possible average elapsed times, which renders it unfeasible to attain every desired time precisely. Consequently, when benchmarking, even if the objective is to achieve a 10-minute average elapsed time, the actual average elapsed time may not be precisely 10 minutes due to this limitation. Allowing the PoW to reach all possible values for the elapsed time ensures greater reliability in the estimation of the 10-minute interval, thereby avoiding the potential for ambiguity in the application of the 10-minute rule.

4.2.2. Adaptation of proof of work modelling to other proofs

In order to guarantee that the elapsed time distribution, which is modelled in the PoW by the geometric distribution $\mathscr{G}(p)$, is identical in the three proofs, $\mathscr{G}(p)$ has been incorporated into the random number generator of PoSW and PoHT. Indeed, the distribution of elapsed time for the PoSW depends only on the random number of iterations required for execution and the one for PoHT depends only on the random number *wait_time*. But random number generators in the nodes produce numbers from a uniform distribution. Therefore, a transformation is required to obtain the same distribution as PoW. Equation 3 defines a transformation function $transform\_prob_p$ function, which converts a uniform distribution into a geometric distribution [42].

$$transform\_prob_p(U) = \left\lceil \frac{ln(1-U)}{ln(1-p)} \right\rceil, U \sim \mathscr{U}[0,1] \tag{3}$$

4.3. Embedded architectures

The three described proofs are embedded within the Trust@TEE.TIME embedded architecture. Proof generation is performed within the TrustZone of the microprocessor, as illustrated in Figure 3. The proofs call hardware and software functions that are described in this subsection.

4.3.1. Proof of work embedded architecture

Figure 4 illustrates the generation of a block for the PoW. The PoW mechanism relies on the *SHA-256* hashing function. To execute *SHA-256*, a hardware crypto-accelerator exclusively accessible from the TrustZone is employed. Once a valid solution to the PoW is found, the header is encapsulated within a transaction and signed using the ECDSA signature function, with the private key securely stored and accessible only within the TrustZone.

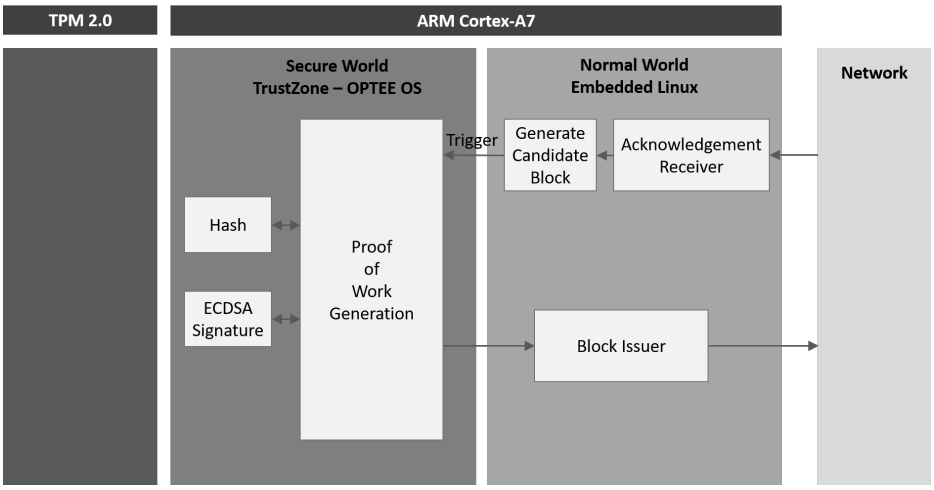


Figure 4. Embedded architecture of PoW on a node with the different functions called.

4.3.2. Proof of sequential work embedded architecture

Figure 5 describes the PoSW generation process, which builds upon the PoW mechanism by incorporating a hardware True Random Number Generator (TRNG). The TRNG, accessible exclusively from the TrustZone, is used to generate a random number at the start of the process. Following this, a sequence of n hashes is performed using the *SHA-256* hashing function. Similar to the PoW, a hardware crypto-accelerator available within the TrustZone is employed for hashing. Once the process is complete, the resulting header is encapsulated in a transaction and signed using the private key inside the TrustZone.

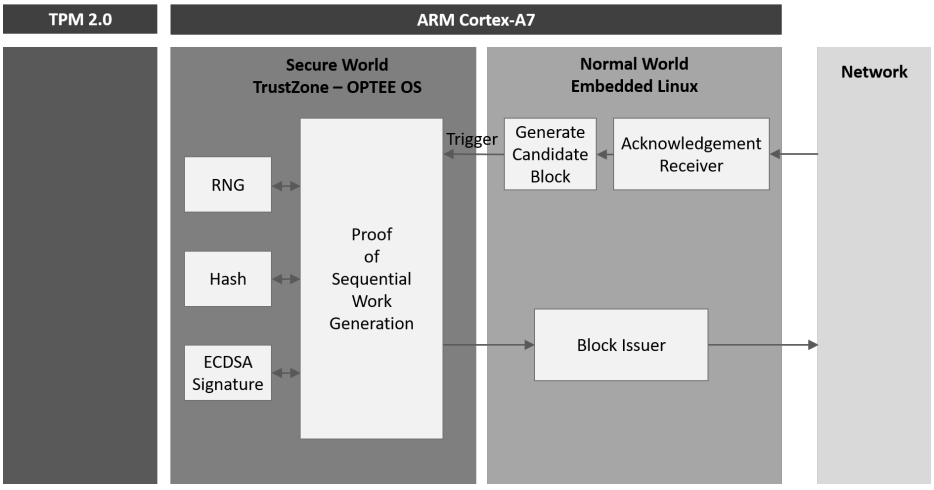


Figure 5. Embedded architecture of PoSW on a node with the different functions called.

4.3.3. Proof of hardware time embedded architecture

Figure 6 illustrates the embedded architecture of the Proof of Hardware Time (PoHT) mechanism, which employs the same procedure for components in the Normal World than the other proofs. The PoHT generation process begins with the creation of a random number using a hardware True Random Number Generator (TRNG) accessed through the TrustZone. Next, a TPM quote attestation is obtained from the Trusted Platform Module (TPM) via a secure Serial Peripheral Interface (SPI). The TrustZone then

initiates a waiting period, after which another TPM quote attestation is performed. These two attestations, along with the wait time, are incorporated into the candidate block. Finally, the header is encapsulated in a transaction and signed using the private key stored in the TrustZone.

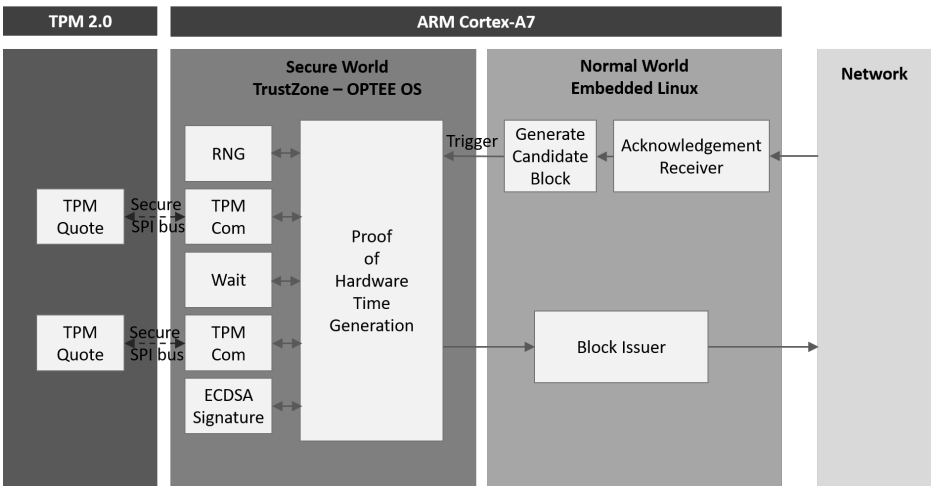


Figure 6. Embedded architecture of PoHT on a node with the different functions called.

To conclude this section, the three proofs are embedded within the TrustZone. A TPM is used in PoHT to provide standardised time attestation. The Normal World is used for communicating with the network. Moreover, the elapsed time distribution in PoW follows a geometric distribution by construction as it has been modelled. In PoSW and PoHT, it depends on the generation of a random number. To achieve the same probability distribution, the $transform\_prob_p$ function is used. Therefore, all three proofs share the same distribution of elapsed time between successive blocks.

5. Experimental results

In our previous study [35], an analysis of the elapsed time and power consumption was conducted, with an average theoretical elapsed time of 13.8 seconds. In this section, the results are enhanced by the addition of further experiments, encompassing several targeted average elapsed time (15 s, 1 min, 5 min and 10 min). A benchmark of the various internal times of the experimental platform is performed to better understand the latency results on the elapsed time.

Figure 7 illustrates the experimental setup based on the Trust@TEE.TIME platform described in Section 3. It consists of 8 nodes, a computer, and a multimeter. Each node is a System on Module (SoM) composed of a STM32MP157F-dk2 evaluation board, including an ARM Cortex-A7 with a Trusted Execution Environment (TEE) ARM TrustZone, and a Trusted Platform Module (TPM) as a daughter board. The ARM Cortex-A7 consists of a Normal World embedding a Linux Operating System (OS), and a Secure World, a TrustZone running an OP-TEE OS [38].

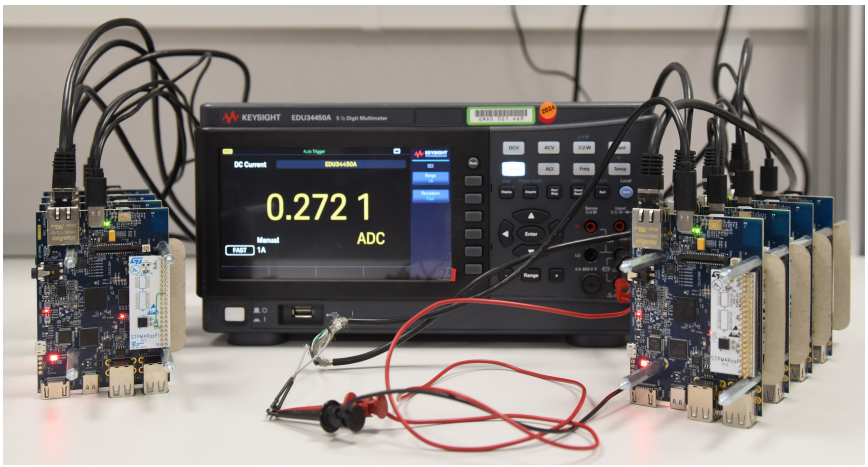


Figure 7. Experimental setup with 8 System on Module composed of STM32MP157f-dk2 evaluation boards, STPM4Raspi daughter boards and the Keysight EDU34450A digital multimeter.

5.1. Platform benchmark

In order to have a better understanding of the experimental results and the influence of the platform configuration on these results, it is essential to perform benchmarking of the timing for all the main functions. Moreover, for PoW and PoSW, it is mandatory to evaluate the hashing capacity of the nodes to ensure the calibration of the elapsed time between two blocks. In this way, we benchmarked: (1) the hash rate for PoW and PoSW, (2) the propagation time of the attestations, (3) the time for the signature of the block and (4) the communication time for retrieving the TPM signatures from the TrustZone.

5.1.1. Hash rate

To calibrate the average elapsed time between two blocks in PoW, the time taken to obtain the nonce and the hash value of the header within the TrustZone is benchmarked. Equation 2 is used to compute the difficulty of achieving the targeted time interval between two successive blocks.

PoW—The process of adjusting the average of the elapsed time distribution to the embedded nodes works as follows: (1) benchmark the time for one iteration of the PoW on the embedded devices, (2) calculate the average elapsed time corresponding to different difficulties and (3) choose the best difficulty n depending on the targeted elapsed time required. The time required for one iteration of the PoW, which comprises one addition, one hash, and one comparison, has been benchmarked in the TrustZone. The resulting value is 8.4 microseconds on nodes of Trust@TEE.TIME. Table 2 shows the mean elapsed time values in the PoW on nodes of Trust@TEE.TIME for different levels of difficulty. As expected with Equation 2, the potential mean elapsed times are discrete, with a factor of two applied for each increase in difficulty. This precludes the possibility of attaining precision for the mean elapsed time by selecting the difficulty level. Thus, for a targeted 15-second interval, the difficulty level of 21 is chosen, resulting in a mean elapsed time of 17.6 seconds. Similarly, for a one-minute interval, the mean elapsed time is 70.5 seconds, for a five-minute interval, it is 282 seconds, and for a ten-minute interval, it is 564 seconds.

PoSW—In the case of the PoSW, two distinct benchmarks are employed. The first test corresponds to

the generation of a random number and the computation of the first hash, while the second test measures the time taken to compute the second hash. One random generation and one hash last 12 milliseconds, and the following hashes last 0.0047 milliseconds. The hashing time in PoSW is shorter than in PoW due to the fact that a 32 bytes message is hashed, in contrast to the larger 80 bytes header utilised in PoW.

Table 2. Mean elapsed time in the Proof of Work on a node of Trust@TEE.TIME for different difficulty levels.

Difficulty n	Mean elapsed time (s)
20	8.8
21	17.6
22	35.2
23	70.5
24	141
25	282
26	564
27	1128

PoHT—In PoHT, devices do not mine or use hashing power. Instead, they generate a random number *wait_time* from a secure hardware True Random Number Generator (TRNG). The TRNG is driven and accessible only from the TrustZone and cannot be altered by the Normal World. The random number is generated from a uniform distribution. Therefore, Equation 3 is used to transform the random number to the according geometric distribution.

5.1.2. Propagation time of the attestation

The propagation time of the attestation can be defined as the interval between the submission of the header to the smart contract and the reception of a receipt emitted by the smart contract. This encompasses the transmission time of the header, its recording within Ganache, and its transmission to the network via an event. It represents the time required for the propagation of the header within the network.

The propagation takes 51 ms for PoW, 152 ms for PoSW and, 234 ms for PoHT. The time required for propagation depends on the proof in question. In the case of PoHT, the header is longer, which requires more time for the smart contract to be processed.

5.1.3. Signature of the header

The header is encapsulated in an Ethereum transaction. The Ethereum signature is performed on the secp256k1 curve [43]. As the signature is reliant solely on the hash of the transaction, we benchmark the signature algorithm for a randomly generated hash. The time of the signature does not depend on the proof considered, as the length of the hash is the same for all proofs.

The signature takes 900 ms on the nodes of the experimental platform, which is significantly longer compared to the time required for a hash computation. The Ethereum ECDSA signature relies on the secp256k1 curve, which is non-standard. Its implementation was carried out within the Trusted Zone (TZ) of the experimental platform, and future work will aim to optimise it for better performance.

5.1.4. Time for retrieving TPM signatures

PoHT requires two communications with the TPM to retrieve the TPM's time measurements. These times are signed by a hardware key within the TPM. The two TPM signed times are recovered from the TPM in 1260 ms.

5.2. Experimental results

5.2.1. Elapsed Time

Figure 8 displays the distributions of the elapsed time between two consecutive blocks in the three implemented proofs for the considered theoretical average elapsed time. The experimental average elapsed times are as follows: for the 15-seconds experiment: 18.9 s for PoW, 19.4 s for PoSW, and 22.5 s for PoHT, for the one-minute experiment 71.1 s for PoW, 72.8 s for PoSW, and 74.1 s for PoHT, for the five-minute experiment: 280 s for PoW, 290.7 s for PoSW, and 305 s for PoHT, and for the ten-minute experiment: 572.7 s for PoW, 569 s for PoSW, and 582.5 s for PoHT. For all the experiments, the elapsed time distribution follows the trend of the geometric laws. To evaluate the adequacy of the observed distribution to the theoretical distribution, a quantile-quantile plot is used in Figure 9. This plot compares the position of quantiles in the observed population with those in the theoretical population. It highlights that the experimental distributions do conform to the modelled distributions in paragraph 4.2. Indeed, the experimental data follow the geometric distribution for the majority of values with a latency. This latency arises due to the observed mean elapsed time deviating from the theoretical mean. The discrepancy is attributed to outliers, namely extreme values where the experimental data diverge from the expected theoretical distribution.

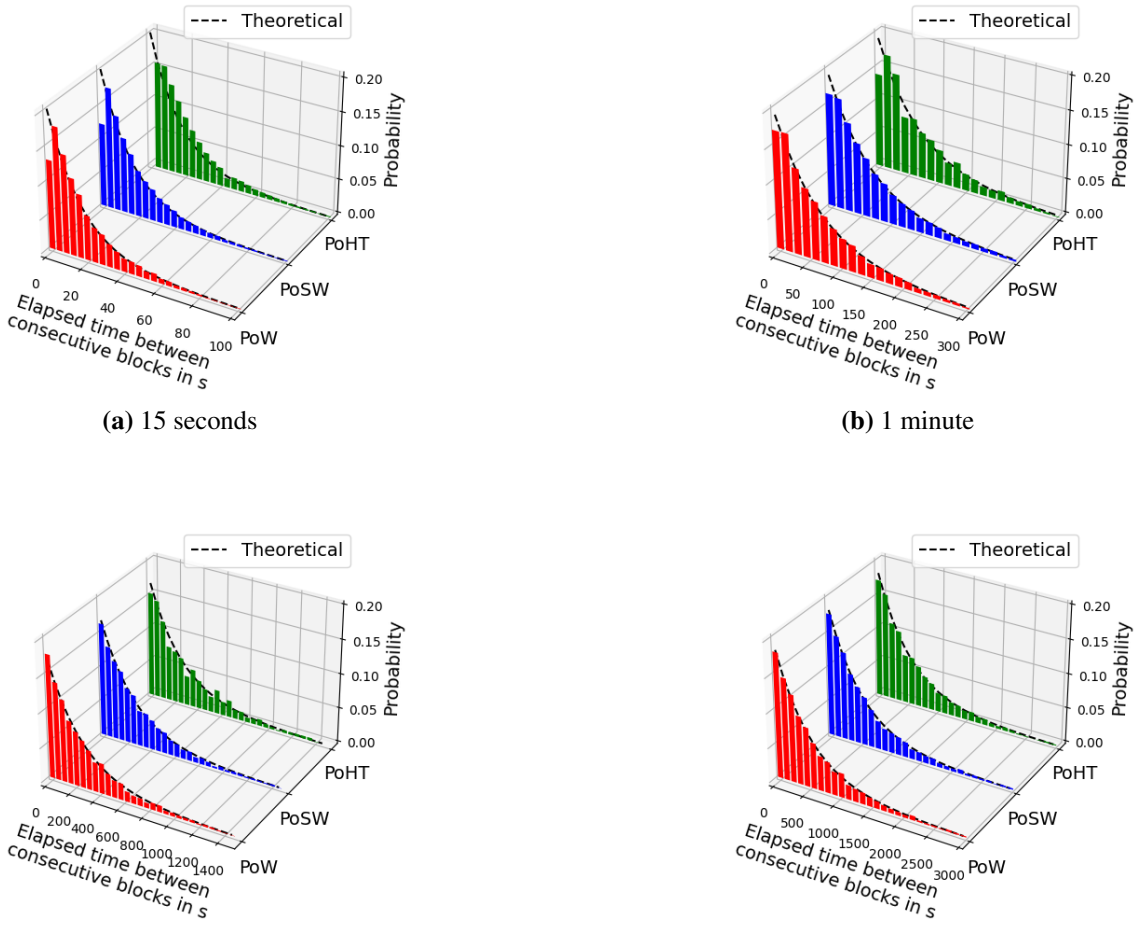
The use of secure hardware makes the proof generation non-parallelisable at the cost of more complex mechanisms involving communication on the SPI bus between the TrustZone and the TPM, and a longer block header. A power consumption comparison is conducted among the three verifiable proofs employing a Keysight EDU34450A digital multimeter. Measurements of intensity are captured at intervals of 0.1s, ensuring a meticulous analysis of the exact moments when computations occur in PoHT. The results are depicted in Figure 10, 11, 12, for a 5V supply voltage, spanning 500 s. The intensity curves shown for PoW and PoSW show sharp variations corresponding to the sending and receiving of block. We observed a higher frequency of such variations for the 15 s interval (Figure 10a and Figure 11a) than for the 10 minutes interval (Figure10d and Figure11d). The upward spikes represent the transmission of transactions to the smart contract, indicating communication over the network. In contrast, the downward spikes correspond to the period of time during which the acknowledgement of the transmitted block is awaited. Additionally, the time interval between spikes is not constant, as the time elapsed between blocks is probabilistic and follows a geometric distribution.

5.2.2. Power consumption

The average power consumption measurement of the SoM are performed during 3 hours, ensuring that enough block generations are taking into account, especially for the 10-min average elapsed time. Under reference conditions when idle, the power consumption of the SoM is 1.33W. When running PoW and

PoSW, the power consumption remains constant at 1.68W. This average power consumption is not dependant of the average elapsed time chosen.

During PoHT, the power consumption stays around the reference power consumption, and rises to 1.68W during the signature process and transfer of the block to the network. Thus, the average power consumption of PoHT is dependent on the average elapsed time chosen. Choosing a longer elapsed time, decreases the power consumption. For 15 s, the power consumption averages at 1.38W, whereas for 10 min it averages at 1.333W.



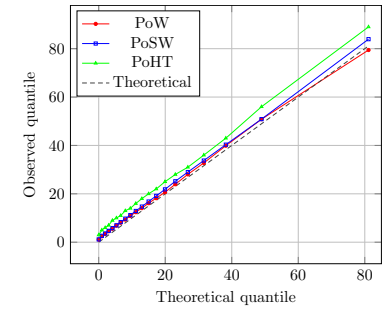
(a) 15 seconds

(b) 1 minute

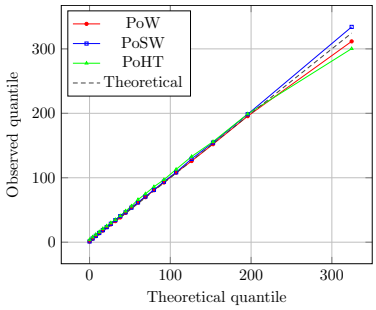
(c) 5 minutes

(d) 10 minutes

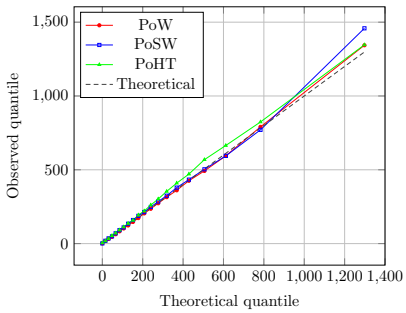
Figure 8. Comparison of the Proof of Work, the Proof of Sequential Work and Proof of Hardware Time conducted on the distribution of elapsed time between two consecutive blocks for an average elapsed time of **(a)** 15 seconds, **(b)** 1 minute, **(c)** 5 minutes and **(d)** 10 minutes.



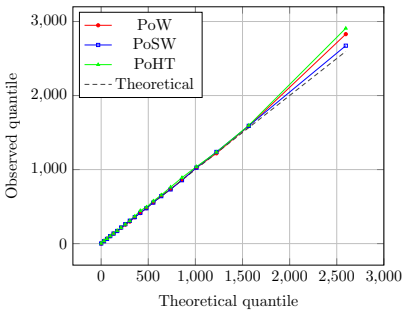
(a) 15 seconds



(b) 1 minute

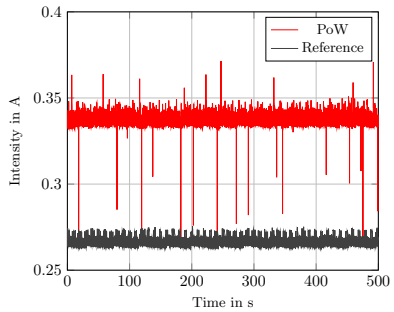


(c) 5 minutes

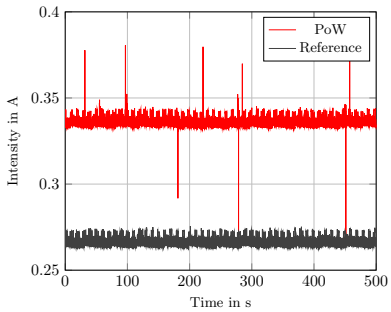


(d) 10 minutes

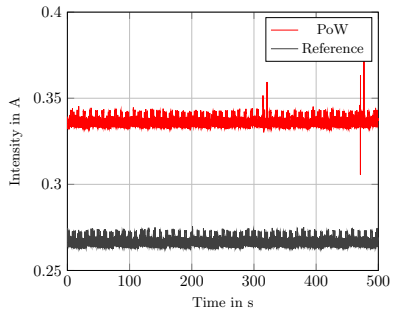
Figure 9. Comparison of the Proof of Work, the Proof of Sequential Work and Proof of Hardware Time conducted on the quantile-quantile plot of the distribution of elapsed time between two consecutive blocks for an average elapsed time of **(a)** 15 seconds, **(b)** 1 minute, **(c)** 5 minutes and **(d)** 10 minutes.



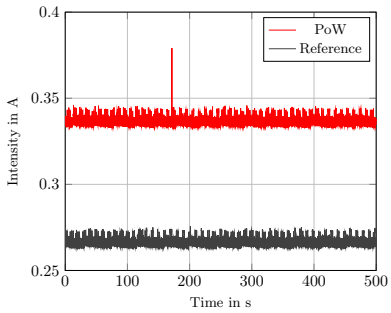
(a) 15 seconds



(b) 1 minute

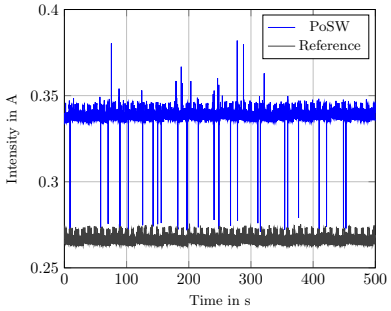


(c) 5 minutes

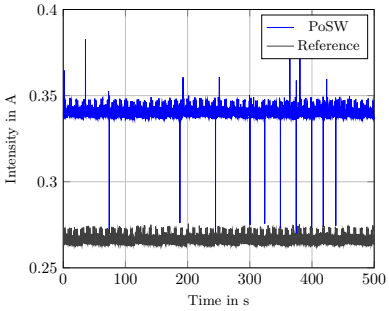


(d) 10 minutes

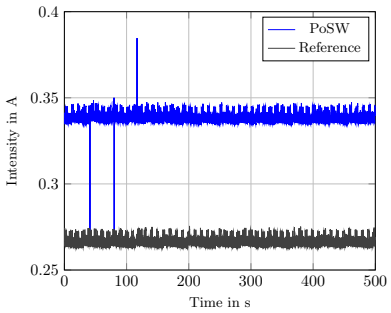
Figure 10. Comparison of the power consumption of the Proof of Work conducted on different average elapsed times between two consecutive blocks: **(a)** 15 seconds, **(b)** 1 minute, **(c)** 5 minutes and **(d)** 10 minutes.



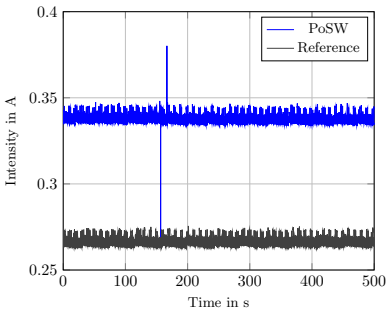
(a) 15 seconds



(b) 1 minute

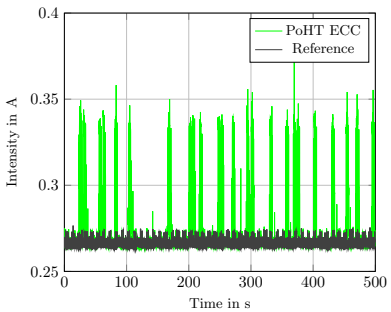


(c) 5 minutes

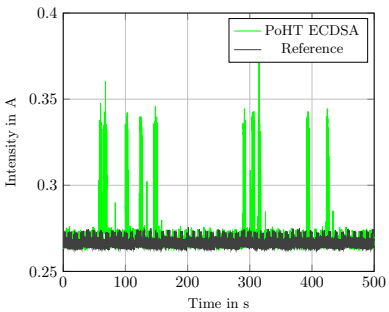


(d) 10 minutes

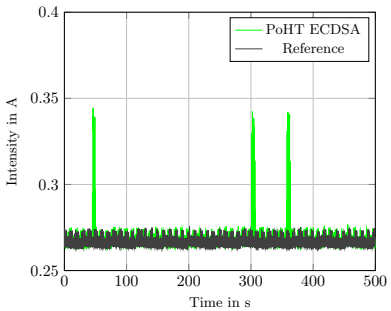
Figure 11. Comparison of the power consumption of the Proof of Sequential Work conducted on different average elapsed times between two consecutive blocks: **(a)** 15 seconds, **(b)** 1 minute, **(c)** 5 minutes and **(d)** 10 minutes.



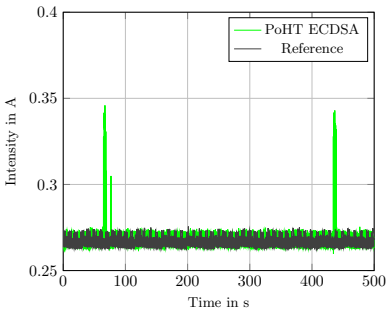
(a) 15 seconds



(b) 1 minute



(c) 5 minutes



(d) 10 minutes

Figure 12. Comparison of the power consumption of Proof of Hardware Time conducted on different average elapsed times between two consecutive blocks: **(a)** 15 seconds, **(b)** 1 minute, **(c)** 5 minutes and **(d)** 10 minutes.

5.3. Discussion

Table 3 presents the experimental results. For the 15-second experiment, the results are consistent with those reported in our previous work [35], except for the latency of PoW. The observed reduction in latency can be attributed to the use of a more efficient PoW algorithm and the significantly faster iteration times in the updated implementation, which allow for a greater degree of precision in the elapsed time. For the other experiments, however, the latency is higher. This increase can be explained by the presence of outliers, which heavily influence the measurement of the experimental average elapsed time. However, in a consensus protocol where only a minimal time is required to determine a winner, these extreme values will not significantly affect the protocol's operation or the delay in proposing a block.

Table 3. Summary of the experimental results based on the mean power consumption and the latency of the implemented Proof of Work (PoW), Proof of Sequential Work (PoSW) and Proof of Hardware Time (PoHT) depending on the average elapsed time.

Average Elapsed Time	Proof Mechanism	Mean Power Consumption	Latency
∼ 15 s	PoW	0.35 W	1.3 s
	PoSW	0.35 W	1.8 s
	PoHT	0.05 W	4.9 s
∼ 1 min	PoW	0.35 W	0.6 s
	PoSW	0.35 W	2.3 s
	PoHT	0.024 W	4.5 s
∼ 5 min	PoW	0.35 W	- 2.1 s
	PoSW	0.35 W	8.7 s
	PoHT	0.009 W	23.04 s
∼ 10 min	PoW	0.35 W	8.7 s
	PoSW	0.35 W	5.0 s
	PoHT	0.003 W	18.4 s

For PoHT, the latency is more significant. This can be explained by the time it takes to communicate with the TPM through the SPI bus and the additional signatures required. The headers used are longer, which also contributes to increase the latency. The latency introduced by PoHT can affect real-world blockchain applications, particularly in vehicular IoT, where system performance is highly sensitive to delays. To make PoHT suitable for such applications, latency must be minimised through optimisations. However, a key advantage of PoHT is its low power consumption, offering a trade-off between energy efficiency and block production rate, which ultimately determines the system's throughput. Additionally, the average elapsed time can be adjusted based on the required transaction rate per second, allowing for adaptive optimisation of energy consumption across the embedded network.

6. Conclusion

In this work, we proposed Trust@TEE.TIME, an experimental platform for comparing proof mechanisms designed to guarantee the elapsed time. The current implementation consists of eight nodes, but its design allows for easy scalability by adding more nodes. This platform is agnostic to the proofs under consideration. We modelled the distribution of elapsed time between consecutive blocks of Proof of Work, Proof of Sequential Work and Proof of Hardware Time and implemented these proofs in

Trust@TEE.TIME. A measurement of the power consumption over the different proofs revealed that the power consumption for PoW and PoSW is constant regardless of the chosen average elapsed time. Conversely, the power consumption of PoHT decreased significantly when a longer average elapsed time was selected, increasing the difference in power consumption between PoW and PoHT from a factor 7 for an interval of 17 s between two blocks, to a factor 117 for an interval of 10 minutes. PoHT generates elapsed time attestations, i.e., proof values, with low power consumption, making it a good candidate for a lightweight blockchain in embedded systems. We showed that communications and signatures induce a latency compared to the model. The latency is larger in PoHT because of the time required to get time attestations provided by the TPM.

For future work, we intend to extend our experimental platform with direct peer-validator communication and to investigate consensus protocols based on Proof of Hardware Time. In addition, we plan to optimise PoHT to reduce the associated delay (e.g., by optimising the signature algorithms). Although our current platform consists of eight nodes, it was designed with scalability in mind, so the number of supported validators can be easily increased to explore larger networks.

Acknowledgments

This work is a collaborative research action that is supported by the French National Research Agency (ANR) in the framework of the “investissements d’avenir” program ANR-10-AIRT-05, irtnanoelec.

Conflicts of interests

The author declare no conflicts of interest.

Author’s contribution

Conceptualization, J.Q., H.C., K.Y. and B.V.; methodology, J.Q., H.C., K.Y. and B.V.; software, J.Q.; validation, J.Q., H.C., K.Y. and B.V.; data curation, J.Q., H.C., K.Y. and B.V.; writing—original draft preparation, J.Q.; writing—review and editing, J.Q., H.C., K.Y. and B.V. All authors have read and agreed to the published version of the manuscript.

References

[1] Nakamoto S. Bitcoin. *A peer-to-peer electronic cash system* 2008, 21260.

[2] Gerrits L, Samuel CN, Kromes R, Verdier F, Glock S, *et al.* Experimental scalability study of consortium blockchains with BFT consensus for IoT automotive use case. In *Proceedings of the 19th ACM Conference on Embedded Networked Sensor Systems*. Coimbra, Portugal, November 15–17, 2021, pp. 492–498.

[3] Chatzigiannis P, Baldimtsi F, Chalkias K. *SoK: Blockchain Light Clients*; Eyal I, Garay J, Eds. Lecture Notes in Computer Science; Cham: Springer, 2022.

[4] Milutinovic M, He W, Wu H, Kanwal M. Proof of luck: An efficient blockchain consensus protocol. In *proceedings of the 1st Workshop on System Software for Trusted Execution*. Trento, Italy, December 12 – 16, 2016 pp. 1–6.

[5] Group TC. TPM 2.0 Library Specification, 2014. Available: https://trustedcomputinggroup.org/resource/tpm-library-specification/ (accessed on 2024-11-27).

[6] Coker G, Guttman J, Loscocco P, Herzog A, Millen J, *et al.* Principles of remote attestation. *Int. J. Inf. Secur.* 2011, 10:63–81.

[7] Anceaume E, Pozzo A, Rieutord T, Tucci-Piergiovanni S. On finality in blockchains. *arXiv preprint arXiv:2012.10172* 2020 .

[8] Deirmentzoglou E, Papakyriakopoulos G, Patsakis C. A survey on long-range attacks for proof of stake protocols. *IEEE access* 2019, 7:28712–28725.

[9] Rivest RL, Shamir A, Wagner DA. Time-lock puzzles and timed-release crypto 1996 .

[10] Boneh D, Bonneau J, Bünz B, Fisch B. Verifiable Delay Functions. *Lect. Notes Comput. Sci.* 2018, 757–788.

[11] Gritti C, Hayek A F, Lafourcade P. Generic Blockchain on Generic Human Behavior. In *SECRYPT 2023*. Rome, Italy, July 10–12, 2023 .

[12] Grollemund P, Lafourcade P, Thiry-Atighehchi K, Tichit A. Proof of behavior. In *The 2nd Tokenomics Conference on Blockchain Economics, Security and Protocols*. Toulouse, France, October, 2020 .

[13] Mahmoody M, Moran T, Vadhan S. Publicly verifiable proofs of sequential work. In *Proceedings of the 4th conference on Innovations in Theoretical Computer Science*. Berkeley, California, USA, January 9 – 12, 2013, pp. 373–388.

[14] Nguyen C, Hoang D, Nguyen D, Niyato D, Nguyen H, *et al.* Proof-of-stake consensus mechanisms for future blockchain networks: fundamentals, applications and opportunities. *IEEE access* 2019 7:85727–85745.

[15] King S, Nadal S. Ppcoin: Peer-to-peer crypto-currency with proof-of-stake. *self-published paper, August* 2012, 19(1).

[16] Li W, Andreina S, Bohli J, Karame G. Securing proof-of-stake blockchain protocols. In *Data Privacy Management, Cryptocurrencies and Blockchain Technology: ESORICS 2017 International Workshops, DPM 2017 and CBT 2017*. Oslo, Norway, September 14-15, 2017, pp. 297–315.

[17] Liu Y, Wang K, Lin Y, Xu W. *LightChain* : A Lightweight Blockchain System for Industrial Internet of Things. *IEEE Trans. Ind. Inform.* 2019, 15(6):3571–3581.

[18] Bai L, Hu M, Liu M, Wang J. BPIIoT: A Light-Weighted Blockchain-Based Platform for Industrial IoT. *IEEE Access* 2019, 7:58381–58393.

[19] Abusalah H, Fuchsbauer G, Gaži P, Klein K. SNACKs: Leveraging Proofs of Sequential Work for Blockchain Light Clients. Cryptology ePrint Archive, Paper 2022/240, 2022. https://eprint.iacr.org/2022/240.

[20] Dorri A, Kanhere S, Jurdak R, Gauravaram P. LSB: A Lightweight Scalable Blockchain for IoT security and anonymity. *J. Parallel Distrib. Comput.* 2019, 134:180–197.

[21] Bada A, Damianou A, Angelopoulos C, Katos V. Towards a Green Blockchain: A Review of Consensus Mechanisms and their Energy Consumption. In *2021 17th International Conference on Distributed Computing in Sensor Systems (DCOSS)*. Pafos, Cyprus, July 14 – 16, 2021, pp.

503–511.

[22] Platt M, Sedlmeir J, Platt D, Xu J, Tasca P, *et al.* The Energy Footprint of Blockchain Consensus Mechanisms Beyond Proof-of-Work. In *2021 IEEE 21st International Conference on Software Quality, Reliability and Security Companion (QRS-C)*. Hainan, China, December, 2021 pp. 1135–1144.

[23] Salimitari M, Chatterjee M. A survey on consensus protocols in blockchain for IoT networks. *arXiv preprint arXiv:1809.05613* 2018 .

[24] Castro M, Liskov B. Practical byzantine fault tolerance. In *OsDI*. February, 1999, vol. 99 pp. 173–186.

[25] Popov S. The tangle. *White paper* 2018, 1(3):30.

[26] IOTA. The Coordinator - PoA Consensus, 2014. Available: https://wiki.iota.org/learn/protocols/coordinator/ (accessed on 2024-03-06).

[27] Baird L, Harmon M, Madsen P. Hedera: A public hashgraph network & governing council. *White Paper* 2019, 1:9–10.

[28] Luckychain. Proof of luck Intel SGX and IPFS based blockchain. Available: https://github.com/luckychain/lucky/ (accessed on 2024-12-06).

[29] Middleton D. Announcing Hyperledger Sawtooth 1.0!, January 30, 2018. Available: https://www.hyperledger.org/blog/2018/01/30/announcing-hyperledger-sawtooth-1-0/ (accessed on 2024-06-03).

[30] Bowman M, Das D, Mandal A, Montgomery H. On elapsed time consensus protocols. In *Progress in Cryptology–INDOCRYPT 2021: 22nd International Conference on Cryptology*. Jaipur, India, December 12 – 15, 2021 pp. 559–583.

[31] Souppaya M, Bartock M, Scarfone K, Dodson D, Carroll D, *et al.* Trusted Cloud: Security Practice Guide for VMware Hybrid Cloud Infrastructure as a Service (IaaS) Environments. *J. Res. Natl. Inst. Stand. Technol.* 2022 .

[32] Johnson S, Scarlata V, Rozas C, Brickell E, Mckeen F, *et al.* Intel software guard extensions: EPID provisioning and attestation services. *White Paper* 2016, 1(1-10):119.

[33] Ling Z, Yan H, Shao X, Luo J, Xu Y, *et al.* Secure boot, trusted boot and remote attestation for ARM TrustZone-based IoT Nodes. *J. Syst. Archit.* 2021, 119:102240.

[34] Ankergård SFJJ, Dushku E, Dragoni N. PERMANENT: Publicly Verifiable Remote Attestation for Internet of Things Through Blockchain. In *Foundations and Practice of Security: 14th International Symposium, FPS 2021*. Paris, France, December 7–10, 2021, pp. 218–234.

[35] Jayet Q, Hennebert C, Kieffer Y, Beroulle V. Embedded Elapsed Time Techniques in Trusted Execution Environment for Lightweight Blockchain. In *2024 IEEE International Conference on Blockchain (Blockchain)*. Copenhagen, Denmark, 2024, pp. 81–88.

[36] Jayet Q, Hennebert C, Kieffer Y, Beroulle V. Securing Elapsed Time for Blockchain: Proof of Hardware Time and some of its Physical Threats. In *2024 27th Euromicro Conference on Digital System Design (DSD)*. Paris, France, 2024 pp. 251–259.

[37] Truffle. Ganache: A tool for creating a local blockchain for fast Ethereum development, 2014.

Available: https://github.com/trufflesuite/ganache/ (accessed on 2024-11-10).

[38] OP-TEE. OP-TEE Documentation. Available: https://optee.readthedocs.io/en/latest/index.html/ (accessed on 2023-11-29).

[39] Wesolowski B. Efficient Verifiable Delay Functions. *J. Cryptol.* 2020, 33(4):2113–2147.

[40] Pietrzak K. Simple verifiable delay functions. In *10th innovations in theoretical computer science conference (itcs 2019)*. 2019 pp. 1–15.

[41] Group TC. TPM 2.0 Keys for Device Identity and Attestation. Available: https://trustedcomputinggroup.org/resource/tpm-2-0-keys-for-device-identity-and-attestation/ (accessed on 2024-03-06).

[42] Gentle JE. Random number generation and Monte Carlo methods, 2003.

[43] Wiki B. Secp256k1. Available: https://en.bitcoin.it/wiki/Secp256k1/ (accessed on 2024-12-06).