Engineering Science & Technology
https://ojs.wiserpub.com/index.php/EST/

UNIVERSAL WISER PUBLISHER

Research Article

Integrating Ethereum and SGX-TEE for Job Execution

Yan Almeida[1], Vladimir Rocha[1*], Carlo Kleber da Silva Rodrigues[1], Arlindo Flavio da Conceição[2]

[1]Center for Mathematics, Computing, and Cognition, Federal University of ABC, São Paulo, Brazil
[2]Institute of Science and technology, Federal University of São Paulo, São Paulo, Brazil
E-mail: vladimir.rocha@ufabc.edu.br

Received: 29 June 2026; **Revised:** 3 August 2026; **Accepted:** 3 August 2026

Abstract: This work proposes an architecture that integrates Ethereum blockchain technology and Trusted Execution Environments (TEEs), focusing on Intel Software Guard Extensions (SGX). The main motivation is to address existing challenges and limitations in on-chain job execution by smart contracts, proposing a reliable method that ensures the integrity and confidentiality of off-chain processed jobs. The proposed architecture is based on a strategic combination of the blockchain's smart contract and the TEEs. The smart contract acts as a decentralized coordinator, responsible for job distribution, monitoring the state, and dynamically allocating resources among TEE machines. In this architecture, the SGX allows the creation of secure enclaves that serve as trusted environments where jobs can be processed in isolation, ensuring protection against external and internal attacks. A prototype implementation was developed to demonstrate the technical feasibility and potential benefits of the proposed architecture. Through experimental testing, limits and bottlenecks were observed for scenarios with varying numbers of TEE machines, revealing a direct proportional growth in response times according to the load, as well as the ability of horizontal scaling to increase system performance.

***Keywords*:** ethereum, blockchain, Software Guard Extensions (SGX), Trusted Execution Environment (TEE), job coordinator

1. Introduction

At the forefront of digital transformation, decentralized Applications (dApps) have reshaped the architecture and operations of modern information systems. Their characteristics—autonomy, traceability, and decentralization—enable a new paradigm in the execution of digital services, ranging from financial transactions to games and social networks [1]. The underlying infrastructure of dApps is the blockchain, popularized by Ether (ETH) cryptocurrency, which represents a new way of recording and validating transactions and the state of applications (through smart contracts) in a secure, transparent, and immutable way [2].

However, blockchain processing faces intrinsic challenges, such as scalability limitations and high transaction costs due to processing or gas consumption, in addition to latency and throughput issues. Thus, on-chain processing can become prohibitive for applications requiring high transaction throughput or those processing large datasets [3]. In this context, off-chain solutions have emerged as promising alternatives to alleviate the load on the blockchain. In such solutions, processing and resource storage are allocated in machines that do not belong to the blockchain. However, many alternatives still struggle to balance security with efficiency—an issue that Trusted Execution Environments (TEEs)

Copyright ©2026 Vladimir Rocha, et al.
DOI: https://doi.org/10.37256/est.81202710397
This is an open-access article distributed under a CC BY license
(Creative Commons Attribution 4.0 International License)
https://creativecommons.org/licenses/by/4.0/

seek to resolve.

The use of TEEs in this context is motivated by the need to establish a verifiable trust boundary for off-chain processing. When moving processing outside the blockchain to improve scalability and reduce gas costs, the execution environment becomes potentially exposed to malicious operators, compromised operating systems, or external attacks. A TEE mitigates these risks by providing hardware-enforced isolation, ensuring that code and data are protected during execution. This guarantees the confidentiality and integrity of sensitive workloads, while still allowing the blockchain to serve as a transparent coordination and verification layer. Therefore, TEEs enable a secure off-chain execution model without sacrificing the trust assumptions required by decentralized applications [4].

TEEs, such as Intel Software Guard Extensions (SGX), offer a secure environment for code execution, isolating critical operations from untrusted parts of the system while ensuring confidentiality and integrity of protected data, even when the operating system itself is compromised [5].

To address job allocation among TEE machines, while maintaining decentralization, this work proposes an architecture that establishes a blockchain smart contract as a job coordinator. The blockchain is used not only for immutable data recording but also to implement logic for job allocation, machine health monitoring, and load redistribution, transforming the solution into a reliable and decentralized execution component.

In summary, this work explores the integration between Ethereum and TEEs using Intel SGX to build a scalable architecture for executing jobs off-chain. The proposed architecture establishes a bridge between the two, where TEEs operate as an additional processing layer capable of interacting with the blockchain efficiently and securely, contributing to discussions on the scalability of dApps execution.

The remainder of this paper is organized as follows. Section 2 presents the background concepts on Ethereum, smart contracts, and Trusted Execution Environments, with emphasis on Intel SGX. Section 3 reviews related work and positions our proposal within existing research. Section 4 describes the proposed architecture, followed by Section 6, which details the deployment environment and tools used. Section 7 reports the experimental results and discusses performance findings. Section 8 outlines the study's limitations, acknowledging current constraints and their implications. Finally, Section 9 concludes the paper and highlights directions for future research.

2. Background

This section provides the fundamental concepts regarding the technologies integrated within this architecture: the Ethereum ecosystem, the mechanics of smart contracts, the requirements of Trusted Execution Environments, and the specific implementation of Intel SGX.

2.1 *Ethereum*

The Ethereum network, proposed by Vitalik Buterin in 2013 and officially deployed in 2015, revolutionized blockchain technology by introducing a programmable layer for decentralized Applications (dApps) [6]. Unlike first-generation blockchains, Ethereum utilizes the Ethereum Virtual Machine (EVM), a Turing-complete environment that enables the execution of complex logic through languages like Solidity. The network's native cryptocurrency, Ether (ETH), serves as the primary incentive mechanism and the unit of account for "gas", which represents the computational effort required for data processing and storage.

Ethereum, as a distributed ledger, is a technology that registers transactions securely and transparently, characterized by [7]:

• Anonymity. Transactions are visible, but the identity of the parties is protected by pseudonyms.

• Immutability. Once a record is added to the blockchain, it becomes extremely difficult to alter.

• Traceability. Each transaction can be traced over time, and all processing and information generated is deterministic.

• Decentralization. The blockchain is distributed among all participants in the network, increasing security and reducing the risk of failures.

The Ethereum network architecture consists of three distinct types of nodes: Full Nodes (responsible for downloading and validating transactions), Lightweight Nodes (optimized for devices with limited resources, validating

transactions by querying full nodes), and Archive Nodes (responsible for maintaining a record of all historical transactions) [8].

2.1.1 *Smart contracts in ethereum*

Smart contracts are self-executing scripts where the terms of an agreement are directly written into lines of code. These contracts act as deterministic state machines; they automatically verify and execute logic when triggered by a transaction. While they can maintain internal states, every write operation to the blockchain's global state incurs financial costs, but read-only operations incur no gas cost when executed off-chain.

The financial cost of on-chain execution creates a bottleneck for data-intensive or processing-intensive jobs. Furthermore, smart contracts are "sandboxed", meaning they cannot natively access external data or perform high-performance computing without breaking the system's deterministic nature [9].

2.2 *Trusted execution environment*

A Trusted Execution Environment (TEE) is a secure processing environment running in an isolated kernel, ensuring the confidentiality and integrity of code and data loaded inside. To be considered "trusted", the environment must satisfy requirements that include spatial separation, sanitization of shared resources, controlled information flow, and failure isolation [4].

In spatial separation, one partition cannot read or modify the contents of another, ensuring that even privileged software (like an OS or hypervisor) cannot inspect the private contents. In sanitization, shared hardware resources cannot be used to leak data. In controlled information flow, communication between partitions must be explicitly allowed. Finally, in failure isolation, a security compromise or crash within one partition must not be propagated to other components of the system.

Another important component present in TEEs is attestation. This is the process of proving to a third party (or a blockchain) that specific software is running on genuine, untampered hardware within a secure enclave [10].

Although Intel SGX is one of the most widely adopted TEE implementations in academic and industrial research, it is not the only available technology. Other hardware-based TEEs include ARM TrustZone, widely used in mobile and embedded devices, and AMD Secure Encrypted Virtualization (SEV), which provides memory encryption for virtual machines. Each of these technologies presents different trust models, performance characteristics, and isolation guarantees. The choice of SGX in this work is therefore not due to exclusivity, but rather to its mature ecosystem, enclave granularity, and compatibility with application-level trusted execution.

2.2.1 *Intel SGX*

Intel Software Guard Extensions (SGX) is a hardware-based TEE implementation that allows applications to execute and run in private regions of memory, known as enclaves. Applications running in SGX are divided into untrusted code (running outside the enclave) and trusted code (running inside the enclave) , with the hardware encrypting the enclave to prevent unauthorized access [11].

However, developing native SGX applications is complex, as it requires specialized SDKs and manual refactoring of system calls. To avoid such complexity, there are libraries, such as Gramine [12], that facilitate deploying applications in enclaves.

Gramine is an open-source Library OS (LibOS) optimized for Intel SGX, which abstracts system calls and provides a complete execution environment for the enclave, eliminating the need for code refactoring in most cases. Gramine allows legacy code to run inside enclaves with minimal changes (which is called a Graminization process).

While advantageous, using Gramine can generate performance overhead, generally between 5% and 20%, due to: slow input and output operations (the need for secure data copying), size and memory limits of the enclaves, and SGX restrictions on thread management [13].

3. Related works

Research on integrating blockchain with TEE has produced a wide range of architectural contributions that directly relate to the problem of using blockchain for coordination to execute jobs in TEEs addressed in this paper. This section reviews several relevant contributions in these domains and how they connect to and motivate the present work. Table 1 brings a comparative summary of these studies.

Table 1. Systematic Comparison of Blockchain-TEE Architectures

Study	Blockchain role & Trust assumptions	Coordination & Scheduling strategy	Performance focus & Characteristics
[14]	Acts as a verification layer and oracle focusing on the retrieval of external data.	Uses a hybrid consensus mechanism with a committee for coordination among TEE nodes.	Achieves off-chain latencies of 1-2 seconds for workloads involving up to 1,000 nodes.
[15]	Manages transaction settlement, access control, and auditability within a data trading ecosystem.	TEEs are used as secure job executors, but coordination latency is not the primary focus.	Focuses on security evaluation rather than throughput, tested on small-scale deployments.
[16]	Used mainly for incentive payments, auditability, and immutability.	The blockchain does not act as a job coordinator, lacking an explicit scheduling or distribution model.	Focuses on computational overhead, showing TEEs do not introduce significant additional latency.
[17]	Acts primarily as a trust anchor and verification mechanism.	TEEs execute predefined and application-specific functions rather than arbitrary coordinated jobs.	Shows execution latencies from hundreds of milliseconds to a few seconds with serialized TEE invocations on 1-5 nodes.
[18]	Serves as an immutable registry for events and policies (e.g., signature verifications).	TEE executes the computation, but without a dedicated on-chain coordination model.	Evaluated purely on small-scale prototypes deployed on a limited number of machines.
[19]	Serves as an immutable audit and automation layer.	Focuses on automating predefined outsourcing workflows.	Focuses on cryptographic and execution overhead on a small number of SGX machines.
[20]	Acts as a coordination layer for managing job execution and incentives for trusted services.	The user is responsible for scheduling the job for arbitrary workloads, relying on reputation-based node management.	Serving jobs for up to 1,000 users introduces approximately 20 seconds of additional latency compared to cloud execution.
Proposed architecture	Acts as a decentralized job coordinator establishing a verifiably trustworthy distributed architecture.	The smart contract itself acts as the explicit job coordinator for arbitrary computational jobs, rather than relying on off-chain consensus or user scheduling.	Enforces a clear separation between on-chain coordination and off-chain execution to address scalability, coordination overhead, and TEE limits.

The authors in [14] present Tora, an architecture that combines blockchain and Trusted Execution Environments (TEEs), specifically Intel SGX, in which the blockchain acts as a verification layer while computation is performed off-chain inside trusted enclaves. Tora is designed for using the blockchain as an oracle, focusing on the retrieval of external data. Coordination among TEE nodes is achieved through a hybrid consensus mechanism with a committee, whereas our approach does not rely on off-chain consensus among executors. In terms of performance, the authors report average off-chain latencies of approximately 1-2 seconds for workloads involving up to 1,000 nodes, which is well below typical blockchain mining times. Similar to the limitations discussed in our work, Tora highlights challenges related to TEE availability, node failures, and the need for additional mechanisms to improve scalability.

The authors in [15] propose a blockchain-based data trading ecosystem that leverages Trusted Execution Environments (TEEs) to perform off-chain data processing. Although TEEs are used as secure job executors, the blockchain manages transaction settlement, access control, and auditability. Performance evaluation focuses on security rather than throughput or coordination latency, and experiments are conducted on small-scale Ethereum and SGX deployments. Limitations discussed are consistent with our work, particularly regarding scalability and non-negligible overhead in blockchain-TEE interaction.

The authors in [16] present FairCs, which integrates blockchain and TEEs in a complementary manner. In this work, the blockchain is mainly used for incentive payments, auditability, and immutability, while TEEs are employed to securely validate the data inside enclaves. Unlike our proposal, the blockchain does not act as a job coordinator, nor does the system implement an explicit scheduling or distribution model for computational jobs across multiple enclaves. The evaluation focuses on computational overhead, showing that the use of TEEs does not introduce significant additional latency. Limitations similar to those discussed in our work include scalability constraints and poor TEE performance. The authors in [17] propose POSE, a blockchain and TEE-based architecture aimed at the trustworthy execution of sensitive computation in trusted off-chain environments. The blockchain is primarily used as a trust anchor and verification mechanism. TEEs execute predefined and application-specific functions. Experimental results indicate execution latencies ranging from hundreds of milliseconds to a few seconds, depending on enclave configuration and cryptographic workload, with experiments conducted on a small number of SGX-enabled machines (between one and five nodes). Similar issues to those identified in our work include the overhead of the enclave and memory limitations, and the performance impact of serialized TEE invocations.

The authors in [18] explore the integration of TEEs with blockchain to enhance secure execution and data protection. The blockchain is mainly employed as an immutable registry for events and policies (e.g., signature verifications), while TEE executes the computation. The evaluation is based on small-scale prototypes deployed on a limited number of machines. Limitations comparable to those discussed in our work include scalability challenges, communication overhead between the blockchain and TEEs, and strong dependence on specific trusted hardware, which constrains large-scale adoption.

The authors in [19] propose Secure and Privacy-Preserving Outsourcing Computing Scheme (SPOCS), a privacy-preserving outsourcing computing scheme that integrates blockchain smart contracts with Trusted Execution Environments (TEEs) to ensure confidentiality and integrity during off-chain computation. The blockchain serves as an immutable audit and automation layer for predefined outsourcing workflows. Performance evaluation focuses on cryptographic and execution overhead, with experiments conducted on a small number of SGX-enabled machines. Similar to the limitations discussed in our work, the system exhibits scalability constraints, attestation overhead, and memory limitations inherent to TEE-based off-chain execution.

The authors in [20] present a decentralized system in which a public blockchain coordinates TEE-enabled computation nodes that provide trusted off-chain execution services. It is closely related to our architecture, as the blockchain is used as a coordination layer for managing job execution and incentives, while TEEs act as job executors. However, the system focuses on pay-as-you-go trusted services and reputation-based node management, but it is the user who schedules the job for arbitrary workloads. Experimental results indicate that serving jobs for up to 1,000 users introduces approximately 20 seconds of additional latency compared to centralized cloud execution, with off-chain computation costs around 20% higher. Experiments are conducted on a limited set of SGX-enabled machines. Similar to our conclusions, the authors highlight trade-offs between decentralization, coordination overhead, and system scalability.

Based on the prior studies, the novelty of the proposed architecture is distinguished in two primary ways:

• Smart Contract as an Active Coordinator: Unlike systems where the blockchain is relegated to a passive verification layer, an audit registry, or an incentive tracker, the proposed architecture establishes the smart contract directly as the primary job coordinator. It handles explicit scheduling and distribution models for arbitrary workloads, bypassing the need for off-chain consensus among executors or bypassing the need for forcing the end-user to schedule jobs manually.

• Architectural Separation of Concerns: By cleanly separating the coordination logic (which remains strictly on-chain) from the execution logic (which operates off-chain inside the TEEs as arbitrary job executors), the design directly targets and addresses the core limitations repeatedly identified in prior works, specifically mitigating coordination overhead, scalability constraints, and TEE resource bottlenecks.

4. Proposed architecture

We propose an architecture that uses the blockchain's smart contract as a system coordinator of jobs. The contract's role is to integrate TEE machines, to ensure job distribution, to monitor the system, and to record results securely and

immutably.

Figure 1 shows the overall architecture, which involves the following components:

a. Ethereum Node (Blockchain): smart contract used to store job requests, define states, control available machines, schedule jobs, monitor health, and reallocate jobs upon failure or timeout.

b. TEE Machine (SGX): Composed of two processes communicating via Transmission Control Protocol (TCP) sockets.

• External Process ("untrusted"): Executed outside the enclave, responsible for all communication with the blockchain (job monitoring, sending heartbeats, retrieving data from a URL) and sending data to the enclave.

• SGX Process ("trusted"): Executed within the enclave (Gramine SGX), its function is to receive data, perform secure job processing, and return the results to the external process.

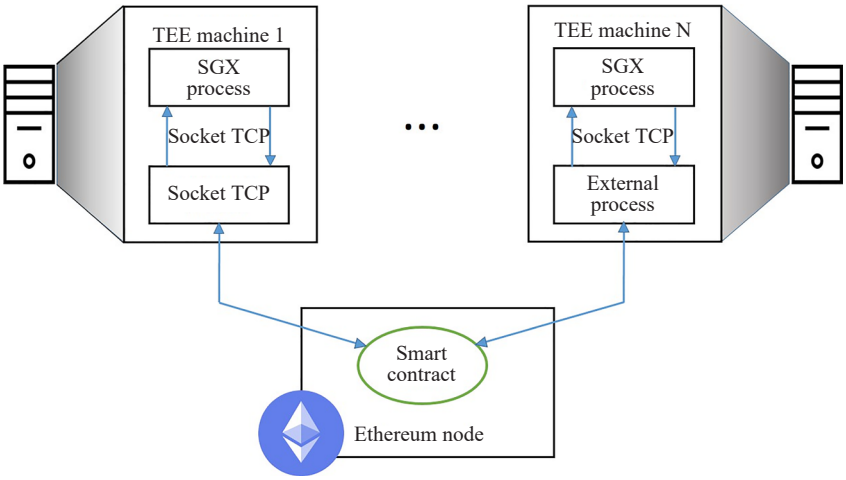


Figure 1. Proposed architecture

In the architecture, the smart contract manages the job lifecycle, with states: WAITING (submitted but not executed yet), PROCESSING (allocated to a TEE machine for execution), and PROCESSED (already executed and results delivered to blockchain). In the current prototype, the allocation of jobs is performed through a Round-Robin algorithm among the available TEE machines. While this simplistic approach serves to validate the baseline coordination logic and system stability, it is resource-agnostic. It does not account for varying workload sizes, individual node health beyond basic heartbeats, or real-time resource utilization (such as available enclave memory or thread saturation). To optimize performance and resource efficiency in a production environment, this architecture allows for the implementation of more advanced scheduling approaches. Future iterations of the architecture could integrate dynamic load balancing, capability-based routing, or reputation-based scoring to allocate jobs based on the real-time capacity and historical performance of each SGX machine.

TEE machines, which are connected with the smart contract, emit heartbeat messages in a period shorter than a timeout. The heartbeat also triggers, in the coordinator, the evaluation of machines whose connection has expired and the reallocation of jobs in the PROCESSING state for too long.

The process of requesting and executing a job occurs through four activities: (i) TEE registration and status confirmation; (ii) User submitting a job and job allocation; (iii) TEE job processing; (iv) User obtaining the results.

Figure 2 shows the first activity, in which the following steps are performed:

1. The TEE machine connects to the smart contract coordinator with ConnectMachine();
2. The smart contract sends to the TEE machine an acknowledgment message;
3. The TEE machine sends an alive message via HeartBeat(), regularly and asynchronously, which allows the smart contract to manage the alive machines.
4. The smart contract sends the connection status to TEE.

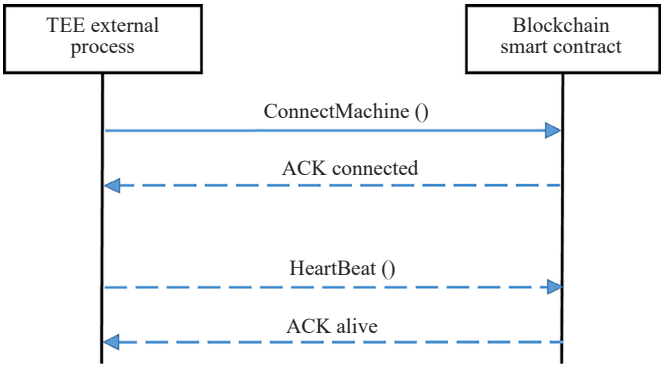


Figure 2. TEE establishes connection and alive activity

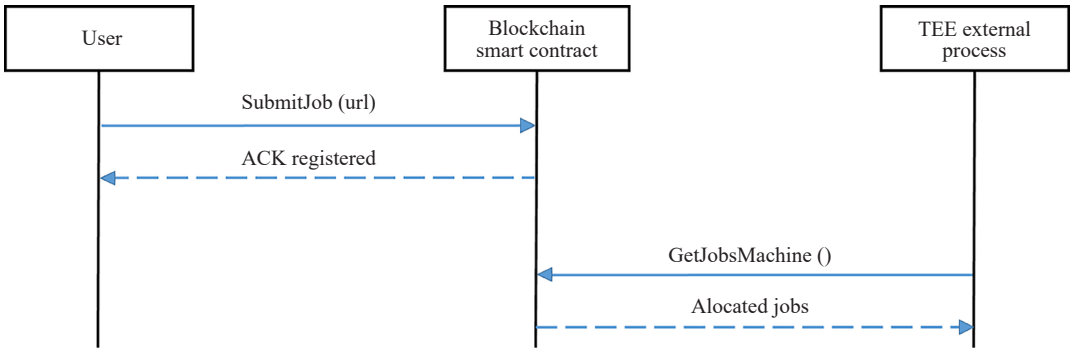


Figure 3. User submits job activity

Figure 3 shows the second activity, in which the following steps are performed:

1. The user submits the job with SubmitJob(url). This URL has the link to the document to be processed by the TEE.

2. The smart contract sends an acknowledgment message to the user and allocates the job to some alive TEE and signals it.

3. The TEE machine (External Process) retrieves allocated jobs from the blockchain with GetJobsMachine() which is delivered by the Smart Contract.

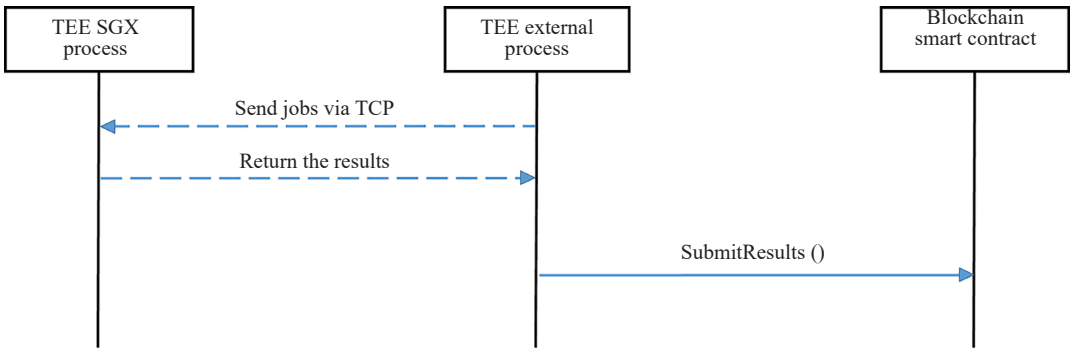


Figure 4. TEE process job activity

Figure 4 shows the third activity, in which the following steps are performed:

1. The TEE machine (SGX Process) obtains the data from the External Process, via TCP, for secure processing,

and processes the Job. In this work, the Job only counts lines on a PDF downloaded from a URL. It is important to note that any code could be executed, but we focus on the technologies integration.

2. The SGX sends the results to the TEE External Process.
3. The TEE External Process submits the result to the blockchain (with Submit Results ());

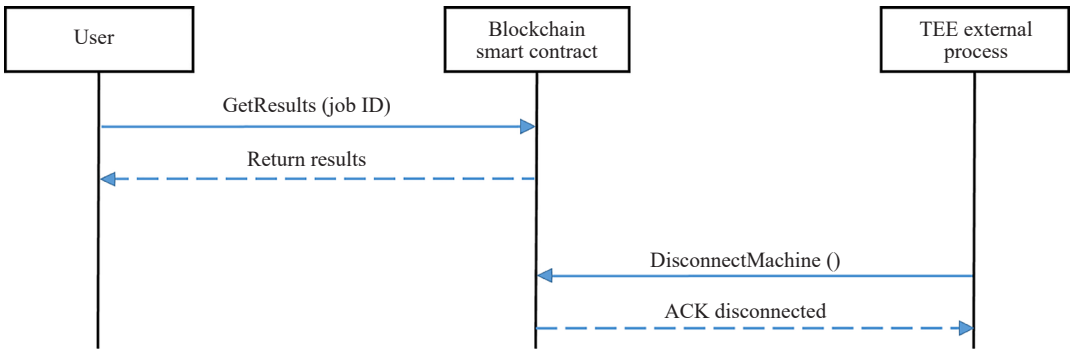


Figure 5. User obtains results activity

Figure 5 shows the fourth activity, in which the following steps are performed:

1. The user retrieves the result from the blockchain smart contract with GetResult (jobId).
2. The TEE machine (External Process) can disconnect from the smart contract to avoid receiving more jobs.

5. Security evaluation and threat model

Although the integration of Ethereum and SGX provides a robust foundation for off-chain execution, a formal evaluation of the security parameters is essential to guarantee the system's reliability. This section defines the formal threat model and discusses the mechanisms implemented to mitigate malicious behaviors, side-channel risks, and protocol-level attacks.

5.1 *Formal threat model and trust as-sumptions*

In the proposed architecture, the trust boundary is strictly delineated between the blockchain and the off-chain execution environment.

• Trusted Components: The Ethereum blockchain is trusted for immutable coordination, job allocation, and state management. The internal hardware components of the Intel SGX processors are trusted to execute the enclaves securely.

• Untrusted Components: The host operating system, hypervisor, external processes communicating with the blockchain, and the network infrastructure are entirely untrusted.

• Adversarial Capabilities: We assume an adversary who can fully compromise the host machine's operating system, intercept or drop network packets between the TEE and the blockchain, and attempt to manipulate the external untrusted process communicating via TCP sockets.

5.2 *Remote attestation and trust validation*

To ensure that jobs are only allocated to genuine and unmodified enclaves, the system relies on remote attestation. Before a TEE machine is authorized to receive jobs in the PROCESSING state, it must generate a cryptographic quote signed by the Intel SGX hardware. This quote proves that the specific software (including the Gramine Library OS and the job processing logic) has been loaded securely and that the hardware is genuine. While the current prototype focuses on the smart contract as a coordinator, a complete production deployment requires this attestation quote to be verified—either via an on-chain verification contract or a trusted off-chain oracle—before the ConnectMachine() function registers

the node as an active participant.

5.3 *Malicious node behavior and enclave scenarios*

A malicious operator hosting a TEE node might attempt to subvert the execution or halt the system.

• Execution Integrity: If the adversary attempts to modify the job pro-cessing logic or tamper with the memory, the SGX hardware will detect the integrity violation and halt the enclave, preventing the submission of fraudulent results to the smart contract.

• Availability Attacks: An adversary may simply disconnect the ma-chine, drop the TCP connection, or prevent the SubmitResults() transaction from reaching the blockchain. The architecture inherently mitigates this malicious node behavior through the smart contract coordinator: if a node fails to deliver a result within the expected timeframe or ceases to send HeartBeat() signals, the con-tract flags the connection as expired and dynamically reallocates the pending jobs using the Round-Robin algorithm to other available, healthy nodes.

5.4 *Rollback and replay protection*

Because the untrusted external process acts as the bridge between the enclave and the blockchain, an attacker could theoretically attempt to intercept network traffic and replay old job requests or rollback the enclave to a previous state.

• Replay Attacks: The architecture defends against replay attacks by utilizing the deterministic and monotonic nature of the Ethereum smart contract. Every job is assigned a unique, immutable jobId. Once a job's state transitions to PROCESSED on the blockchain, any subsequent attempt to submit results for that same jobId will be automatically rejected by the smart contract's state validation logic.

• Rollback Attacks: While SGX enclaves lack native persistent trusted time or storage, the continuous synchronization with the blockchain acts as a secure timestamping mechanism. The smart contract coordinates the absolute state; therefore, rolling back the local state of a TEE machine only results in a localized denial-of-service, which the coordinator resolves via job reallocation.

5.5 *Side-channel risks*

While Intel SGX provides strong spatial separation and memory encryption, TEEs are historically susceptible to side-channel attacks, such as cache timing or page-fault monitoring, where a compromised host OS observes access patterns to infer confidential data. Because our archi-tecture utilizes Gramine to execute legacy code without refactoring, native application-level defenses (such as constant-time execution) are not guaranteed unless explicitly implemented in the underlying payload. Within the context of this architecture, side-channel vulnerabilities represent an accepted risk inherent to the hardware generation. However, the system mitigates the impact of prolonged targeted attacks by allowing rapid job reallocation and horizontal scaling across multiple, potentially distinct, TEE providers, limiting the time an adversary has to monitor a single workload execution.

6. Methods

To test the architecture, it is necessary to understand the infrastructure, the software tools, and the blockchain configuration deployed in the cloud environment.

6.1 *Deployment infrastructure*

The infrastructure consisted of:

• SGX Nodes (Azure cloud platform): Virtual machines of type Standard_DC1ds_v3 (with SGX support) [21], each running Docker containers for both the SGX Process (via Gramine) and the External Process (with Python), communicating through TCP sockets.

• Ethereum Node (AWS cloud platform): An EC2 virtual machine [22] running a Docker container [23] with

Ganache [24], a Node.js-based tool for simulating an Ethereum network.

The bootstrap process for the SGX nodes included installation of the SGX driver, graminization of the Docker containers to enable job execution in the enclave, and launching the containers.

6.2 *Tools used*

The tools used for deploying the architecture are:

• Infrastructure as Code: Terraform [25] was used for deploying the components in Azure TEE machines and AWS blockchain node.

• Containerization: Docker and Gramine Secure Containers (GSC) was used for executing the code associated to jobs in the enclave.

• Programming Language and Libraries: Python 3.8 with libraries such as web3 [26] (Ethereum connection), hdwallet [27] (account generation), and requests (data fetching). The SGX Process used PyPDF2 [28] for job processing (character counting).

• Blockchain Simulation: Node with Ganache (version 7.9.2) was used and configured with parameters close to those of the real Ethereum network (15-second block mining time and gas limits), ensuring realistic and reproducible experimental conditions, as explained in the next section.

• Observability: NewRelic [29] was used to collect metrics and execution traces, enabling analysis of execution time and resource consumption.

6.3 *Blockchain configuration*

To verify that the environment provided by Ganache is sufficiently similar to official Ethereum networks, the contract was deployed on the official Sepholia [30] testnet and compared against Ganache. To validate and isolate the impact of block mining latency on the system's final results, a 1-kilobyte payload was utilized with 100 samples at a rate of 1 RPS (requests per second).

Upon execution, it was observed that Ganache yielded return times with a mean of 46.02 seconds and a standard deviation of 5.58 seconds, while Sepolia showed a mean of 44.62 seconds with a standard deviation of 7.67 seconds. A T-test performed on the data resulted in a significance level of 0.142, which exceeds the common threshold of 0.05, indicating that there is no evidence of a statistically significant difference between the means of the two tests (Sepholia and Ganache). Also, in both tests, identical execution times were observed within the TEE, reinforcing the hypothesis that the configurations assigned to Ganache provide a valid approximation of a real Ethereum network.

In this context, the experiments shown in the next section use Ganache, which will allow us to test the Blockchain behavior and performance without considering the gas required to execute the transactions. This finding increases the confidence in the overall results, particularly since the executions presented in the experiments require a substantial amount of gas, rendering direct large-scale testing on Sepolia unfeasible.

6.4 *Source code*

The code to deploy the architecture and reproduce the experiments could be found in https://github.com/YanAlmeida/projeto-azure-scone-ethereum.

7. Experimental results

The architecture was subjected to stress tests using:

• Request payloads ranging from 1 Kilobyte to 1 Megabyte to validate operational limits.

• Three rates of Requests per Second (RPS), specifically 1, 4, and 6 RPS, to validate execution behavior.

• Three scenarios, specifically with 1, 2, and 3 TEE machines, to evaluate the horizontal scalability.

The metric analyzed for these three scenarios was the end-to-end latency, which corresponds to the time passed, in seconds, between when the user submits a Job request and obtains a results response.

In each scenario, we present two plots. The first shows the request payload from 1 kilobyte to 100 kilobytes (low-to-medium size payload), and the second shows the request payload from 100 kilobytes to 1 megabyte (medium-to-large size payload). Y-axis of the plot exhibits the end-to-end latency, as mentioned before.

7.1 *Scenario 1: One SGX machine*

For low-size payload (1 kilobyte to 10 kilobytes), Figure 6 shows almost a constant end-to-end latency for different RPS. This behavior occurs because, for low-size payloads and lower request rates, neither the blockchain nor the TEE reaches its processing saturation limit. In other words, the block mining interval is sufficient to confirm transactions without queue accumulation, and the enclave's thread and memory resources can process incoming jobs without contention. As a result, the system operates in a stable region where latency remains approximately constant, since no internal bottleneck is triggered in either the coordination layer (blockchain) or the execution layer (TEE).

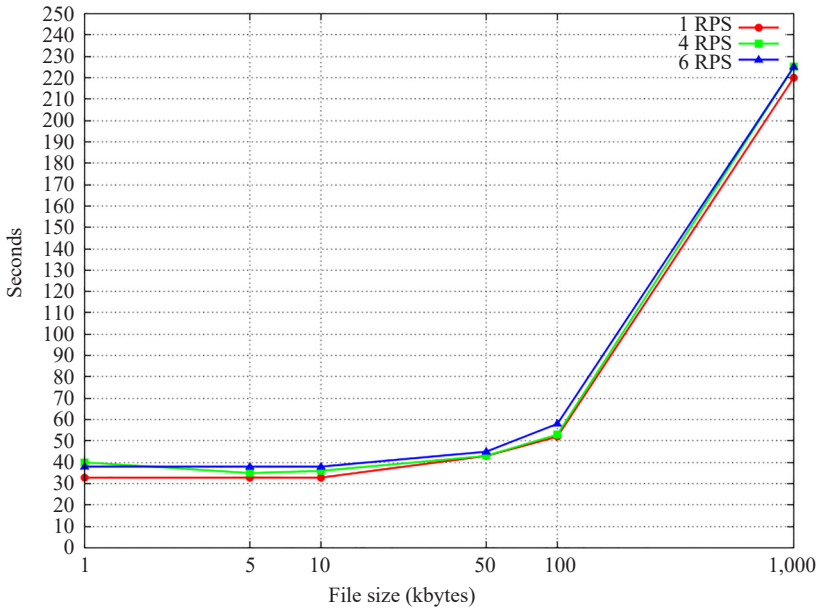


Figure 6. Scenario 1: Latency as a function of payload size

For medium-size payload (10 kilobytes to 100 kilobytes), the figure shows a increasing end-to-end latency for Scenario 1, mainly due to blockchain and thread saturation, as explained below, which limits the architecture's performance even at lower request per second (RPS) rates. The higher end-to-end latency is a consequence of two issues: (a) a sign of thread saturation behavior observed in the TEE execution beyond 1 RPS, caused by the asynchronous jobs; (b) the bottleneck generated by the blockchain and the block mining time, which causes delays in operations related to job state changes and result return.

For large-size payloads (100 kilobytes to 1 megabyte), the same crescent behavior of medium-size payloads is shown, but note that the time at 1 megabyte, which is near 225 seconds, is impractical for almost all dapps. Finally, for a payload of 5 megabytes, an absence of data is ob-served (not shown in the graph) because the thread allocation capacity in TEE is exceeded, preventing it from continuing to process jobs.

7.2 *Scenario 2: Two SGX machines*

Although Figure 7 shows the same behavior as figures of Scenario 1, it is important to explain the differences from the perspective of RPS curves. For lower RPS rates (mainly 1 RPS), the variation in the number of SGX machines (from one to two) shows that horizontal scalability reduces end-to-end latency time for all payload sizes (low, medium,

and large). This is due to the division of the workload, which increases throughput and decreases latency, avoiding the saturation of the individual machine resources.

For RPS rates that exceed the total processing throughput of the architecture (4 and 6 RPS), they present almost the same values as Scenario 1, also due to the two performance issues: thread saturation and block mining time. The only difference is that, in Scenario 2, the constant end-to-end latency finishes at 5 kilobytes, instead of 10 kilobytes of Scenario 1.

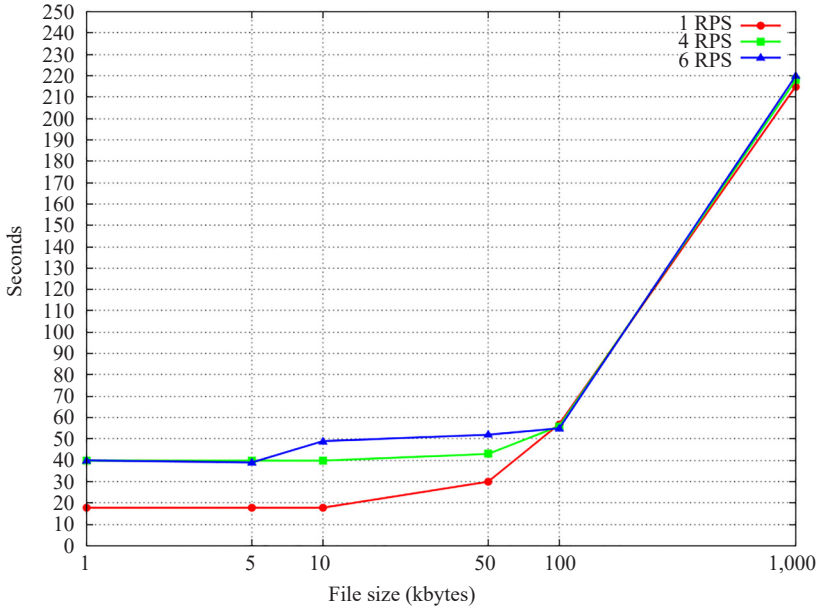


Figure 7. Scenario 2: Latency as a function of payload size

7.3 *Scenario 3: three SGX machines*

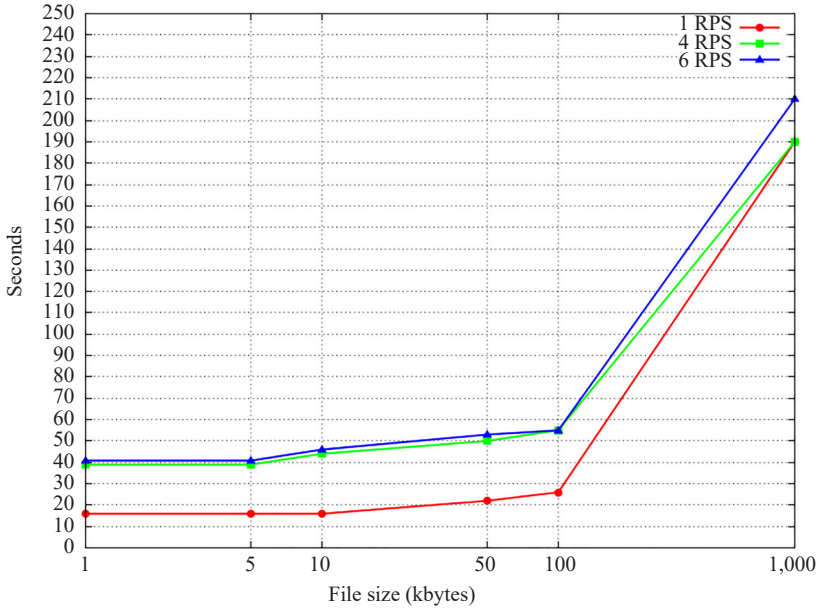


Figure 8. Scenario 3: Latency as a function of payload size

As seen in previous scenarios, Figure 8 also shows the same curve behavior.

For lower RPS rates (mainly 1 RPS), using three SGX machines reduces end-to-end latency time for all payload sizes, showing that horizontal scalability is important to improve system performance.

For RPS rates above (4 and 6 RPS), they present reduced values if compared with Scenarios 1 and 2. As seen in the previous scenario, the constant end-to-end latency finishes before 10 kilobytes of Scenario 1, that is, at 5 kilobytes.

7.4 *Discussion*

Increasing the number of SGX machines (horizontal scalability) offers benefits by decreasing end-to-end latency for lower RPS rates. Table 2 shows this decrease for 1 RPS. Note that, in almost all payloads, the end-to-end latency decreases from scenario 1 to scenario 3. For example, for 1 kilobyte, the latency decreases from 33 seconds with 1 SGX machine to 16.71 seconds with 2 SGX machines, and to 16.64 seconds with 3 SGX machines.

Table 2. Mean end-to-end latency (in seconds) for 1 RPS in all scenarios

Payload	Scenario 1	Scenario 2	Scenario 3
1 KB	33.00 (± 1.05)	16.71 (± 1.47)	16.64 (± 1.79)
5 KB	33.14 (± 1.50)	16.80 (± 1.49)	16.60 (± 1.54)
10 KB	33.17 (± 1.48)	16.59 (± 1.12)	16.62 (± 1.32)
50 KB	42.79 (± 5.21)	30.35 (± 6.64)	21.67 (± 9.95)
100 KB	52.60 (± 5.30)	52.49 (± 5.23)	25.65 (± 9.46)
1 MB	221.00 (± 7.50)	213.82 (± 8.80)	190.00 (± 12.79)

Note also that there are several values with approximately 16 seconds. This value corresponds to the lower limit imposed by blockchain block mining time, which limits the end-to-end latency.

These values confirm that the blockchain dependency remains a fundamental bottleneck for off-chain execution architectures. Because every job submission and result retrieval requires an on-chain state transition, the block mining time imposes a strict lower bound on latency, becoming the dominant performance constraint regardless of the computational power or horizontal scaling of the TEE layer. To mitigate this limitation without sacrificing decentralization, future deployments could integrate Layer-2 scaling solutions, such as Optimistic Rollups or Zero-Knowledge (ZK) Rollups, which aggregate off-chain transac-tions to drastically reduce on-chain congestion and settlement times. Alternatively, deploying the smart contract coordinator on blockchain configurations with faster consensus mechanisms—such as Proof-of-Authority (PoA) networks or high-throughput Layer-1 chains—would significantly minimize this coordination latency.

With respect to throughput, the experiments show that the end-to-end time is reduced with more SGX machines, but increases if more RPS are added. For example, if the payload is 5 kilobytes, 1 RPS takes 33 seconds, 4 RPS takes 36 seconds, and 6 RPS takes 38 seconds. It is important to note that, even with these time increments, they are not proportional, showing that the scalability is maintained.

Note that the higher values obtained confirm that the dependence on computational resources (number of threads, asynchronous jobs), and the Blockchain are the main architecture limitations. These limitations, as explained before, occurs by the enclave's processing capacity and overhead for larger payloads (above 100 kilobytes), and the blockchain mining time used when a user submits the job and when the SGX machine sends the results.

Finally, it is important to frame these scalability conclusions within the context of the tested environment. While the experiments demonstrate proportional performance improvements and confirm the architecture's fundamental ability to distribute jobs horizontally, the current deployment is limited to a maximum of three SGX nodes. Therefore, these results serve as an initial validation of the smart contract co-ordination logic rather than a definitive demonstration of large-scale deployment capabilities or network-wide scalability.

8. Limitations

While the proposed architecture demonstrates the feasibility and benefits of integrating blockchain coordination with SGX-based Trusted Execution Environments, some limitations should be acknowledged as follows. First, the experimental workload used in the evaluation is intentionally simple, focusing on PDF processing jobs. This choice was made to isolate the integration aspects between the blockchain and TEEs and to ensure reproducibility of the experiments. However, we acknowledge that PDF line-counting does not adequately represent realistic, computationally demanding workloads for decentralized applications. Consequently, these results should not be directly generalized to broader cloud or data-intensive workloads without further evidence. Future work will extend the evaluation to more complex scenarios—such as machine learning inference, confidential data analytics, or cryptographic processing—to demonstrate the practical value of the architecture under heavy computational stress. Although it does not represent the full complexity of real-world applications, it provides a controlled baseline that clearly highlights system behavior, bottlenecks, and coordination overhead. Future work will extend the evaluation to more computationally intensive and heterogeneous workloads to better assess performance in practical scenarios. Second, the deployment scale is relatively small, involving up to three SGX machines and a controlled blockchain environment. This setup allowed precise observation of system dynamics and performance limitations, particularly those related to mining latency and enclave saturation. Nevertheless, the scalability conclusions drawn here serve only as an initial validation. Evaluating larger deployments, either through physical testing with a significantly higher number of nodes or through analytical modeling, is necessary to provide stronger evidence regarding coordination overhead and fault tolerance in realistic, large-scale environments. In this context, large-scale deployments may introduce additional challenges such as network variability, fault tolerance, and resource heterogeneity. Expanding the infrastructure to a larger number of nodes and more realistic network conditions is planned as future work. Third, although the architecture leverages TEEs to enhance confidentiality and integrity, the paper does not provide an in-depth analysis of advanced security threats, such as side-channel attacks or detailed adversarial models. The primary goal of this work was to validate the architectural integration and its performance implications. A more comprehensive security evaluation, including formal threat modeling and mitigation strategies, is an important direction for future research. Overall, these limitations reflect deliberate design choices that enabled a focused validation of the proposed architecture. Addressing these aspects in future studies will further strengthen the robustness, scalability, and security understanding of blockchain-TEE integrated systems.

9. Conclusions and future work

This work aimed to design and evaluate a distributed architecture that integrates blockchain coordination with Trusted Execution Environments, specifically Intel SGX, to enable secure and scalable off-chain job execution. The chosen methodology combined architectural design, prototype implementation, and experimental evaluation under controlled conditions, varying payload sizes, request rates, and the number of TEE nodes.

Overall, the results demonstrate that the proposed architecture is technically feasible and capable of improving performance through horizontal scalability, within our initial validation parameters, particularly at lower request rates. Particularly, the experiments confirmed that distributing workloads across multiple SGX machines reduces end-to-end latency, while also revealing practical bottlenecks related to enclave resource limits and blockchain mining time. These findings highlight the trade-offs between decentralization, security, and performance that characterize blockchain-TEE integrations.

Beyond the experimental outcomes, our main contribution thus lies in proposing a clear separation between coordination and execution layers, where smart contracts act as decentralized schedulers and TEEs serve as trusted computation nodes. This design advances the discus-sion on how to build trustworthy distributed systems that maintain data integrity and confidentiality without sacrificing scalability, offering a practical reference architecture for future decentralized applications.

Looking ahead, we consider that future work must explore broader evaluation dimensions to validate the architecture at scale. These directions include blockchain cost analysis, testing more diverse and demanding application scenarios (such as federated machine learning models or high-throughput analytics), optimizing the on-chain

coordination mech-anisms, and conducting a deeper analysis of operational behaviors by expanding the infrastructure to a significantly larger number of heterogeneous SGX nodes. Furthermore, to address the identified bottlenecks, future iterations will focus on optimizing the on-chain coordination mechanisms through resource-aware scheduling algorithms and investi-gating the integration of Layer-2 scaling solutions to reduce blockchain latency. These directions seek to consolidate the proposed approach as a robust foundation for secure and efficient off-chain computation in decentralized ecosystems.

The integration between the Ethereum blockchain and TEEs, using Intel SGX, has proven to be a promising strategy for the secure and efficient execution of jobs, using the smart contract as a decentralized coordinator. Consistent with existing systems, the proposed architecture reinforces the confidentiality and integrity of data (which is processed in the TEE), while indicating the potential to maintain the scalability required for modern decentralized applications. Although significant challenges remain to be overcome, such as performance optimization, resource utilization, and security, this work offers a foundation for future research and developments in the integration between blockchain and TEEs, which are becoming increasingly relevant.

Author contributions

Conceptualization: V.R., C.K.; methodology: V.R., Y.A.; software: Y.A.; validation: Y.A., V.R., A.C.; investigation: Y.A.; writing—original draft preparation: Y.A., V.R., C.K.; writing—review and editing: V.R., C.K., A.C.; project administration: V.R. All authors have read and agreed to the published version of the manuscript.

Acknowledgments

This work is a collaborative research action with the IAsmin Platform: The Evolution of Brazilian Industry to Standard 4.0. Fapesp research grant number: 20/09850-0.

Conflict of interest

The authors declare no potential conflict of interests.

References

[1] P. Zheng, Z. Jiang, J. Wu, and Z. Zheng, "Blockchain-based decentralized application: A survey," *IEEE Open Journal of the Computer Society*, vol. 4, pp. 121-133, 2023.

[2] W. Cai, Z. Wang, J. B. Ernst, Z. Hong, C. Feng, and V. C. M. Leung, "Decentralized applications: The blockchain-empowered software system," *IEEE Access*, vol. 6, pp. 53019-53033, 2018.

[3] I. S. Rao, M. L. M. Kiah, M. M. Hameed, and Z. A. Memon, "Scalability of blockchain: A comprehensive review and future research direction," *Cluster Computing*, vol. 27, no. 5, pp. 5547-5570, 2024.

[4] M. Sabt, M. Achemlal, and A. Bouabdallah, "Trusted execution environment: What it is, and what it is not," in 2015 IEEE Trustcom/BigDataSE/ISPA, 2015, pp. 57-64.

[5] S. Chakrabarti, T. Knauth, D. Kuvaiskii, M. Steiner, and M. Vij, "Trusted execution environment with Intel SGX," in *Responsible Genomic Data Sharing*. Amsterdam, The Netherlands: Elsevier, 2020, pp. 161-190.

[6] V. Buterin and G. Wood, "Ethereum whitepaper," *GitHub Repository*, 2013. [Online]. Available: https://ethereum.org/content/whitepaper/whitepaper-pdf/Ethereum_Whitepaper_-_Buterin_2014.pdf. [Accessed Jul. 17, 2026].

[7] M. J. Fernández-Iglesias, C. Delgado von Eitzen, and L. Anido-Rifón, "Efficient traceability systems with smart contracts: Balancing on-chain and off-chain data storage for enhanced scalability and privacy," *Applied Sciences*, vol. 14, no. 23, pp. 11078, 2024.

[8] H. Feng, Y. Hu, Y. Kou, R. Li, J. Zhu, L. Wu, and Y. Zhou, "SlimArchive: A lightweight architecture for Ethereum archive nodes," in USENIX Annual Technical Conference, 2024, pp. 1257-1272.

[9] A. M. Antonopoulos and G. Wood, *Mastering Ethereum: Building Smart Contracts and DApps*. Sebastopol, CA, USA: O'Reilly Media, 2018.
[10] J. Ménétrey, C. Göttel, A. Khurshid, M. Pasin, P. Felber, V. Schiavoni, and S. Raza, "Attestation mechanisms for trusted execution environments demystified," in IFIP International Conference on Distributed Applications and Interoperable Systems, 2022, pp. 95-113.
[11] W. Zheng, Y. Wu, X. Wu, C. Feng, Y. Sui, X. Luo, and Y. Zhou, "A survey of Intel SGX and its applications," *Frontiers of Computer Science*, vol. 15, no. 3, pp. 153808, 2021
[12] Gramine Project, "Gramine secure container (GSC) documentation," 2025. [Online]. Available: https://gramine.readthedocs.io/projects/gsc/en/latest/. [Accessed Dec. 12, 2025].
[13] D. Kuvaiskii, "Developers guide to Gramine open-source Lib OS for running unmodified Linux apps with Intel SGX," *Microsoft Azure*, 2022. [Online]. Available: https://techcommunity.microsoft.com/blog/azureconfidentialcomputingblog/developers-guide-to-gramine-open-source-lib-os-for-running-unmodified-linux-apps/3645841. [Accessed Jan. 17, 2026].
[14] L. Chen, R. Yuan, and Y. Xia, "Tora: A trusted blockchain oracle based on a decentralized TEE network," in 2021 IEEE International Conference on Joint Cloud Computing (JCC), 2021, pp. 28-33.
[15] W. Dai, C. Dai, K.-K. R. Choo, C. Cui, D. Zou, and H. Jin, "SDTE: A secure blockchain-based data trading ecosystem," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 725-737, 2019.
[16] Y. Liang, Y. Li, and B.-S. Shin, "FairCs—Blockchain-based fair crowdsensing scheme using trusted execution environment," *Sensors*, vol. 20, no. 11, pp. 3172, 2020.
[17] T. Frassetto, P. Jauernig, D. Koisser, D. Kretzler, B. Schlosser, S. Faust, and A.-R. Sadeghi, "POSE: Practical off-chain smart contract execution," in Annual Network and Distributed System Security Symposium (NDSS 2023), 2023, pp. 23118.
[18] T. Peng, K. Guan, and J. Liu, "A privacy-preserving mobile crowdsensing scheme based on blockchain and trusted execution environment," *IEICE Transactions on Information and Systems*, vol. 105, no. 2, pp. 215-226, 2022.
[19] Z. Liu, C. Hu, R. Li, T. Xiang, X. Li, J. Yu, and H. Xia, "A privacy-preserving outsourcing computing scheme based on secure trusted environment," *IEEE Transactions on Cloud Computing*, vol. 11, no. 3, pp. 2325-2336, 2022.
[20] Y. Liang, Y. Li, and B.-S. Shin, "A blockchain-based platform for decentralized trusted computing," *Peer-to-Peer Networking and Applications*, vol. 17, no. 3, pp. 1499-1513, 2024.
[21] Azure Virtual Machines, "Dcdsv3 sizes series," *Microsoft Azure Dcdsv3 Sizes Series*, 2025. [Online]. Available: https://learn.microsoft.com/en-us/azure/virtual-machines/sizes/general-purpose/dcdsv3-series. [Accessed Dec. 17, 2025].
[22] Amazon Web Services, "Amazon EC2: Secure and resizable compute capacity for virtually any workload," *Amazon EC2*, 2025. [Online]. Available: https://aws.amazon.com/ec2/. [Accessed Set. 18, 2025].
[23] Docker Inc., "Docker: Accelerated container application development," *Docker*, 2025. [Online]. Available: https://www.docker.com/. [Accessed Set. 18, 2025].
[24] ConsenSys Software, "Ganache: One-click blockchain," *Ganache*, 2025. [Online]. Available: https://archive.trufflesuite.com/ganache/. [Accessed Oct. 22, 2025].
[25] HashiCorp, "Terraform," *Terraform*, 2025. [Online]. Available: https://developer.hashicorp.com/terraform/. [Accessed Dec. 12, 2025].
[26] Ethereum Foundation, "web3.py: A Python library for interacting with Ethereum," *web3.py documentation*, 2025. [Online]. Available: https://web3py.readthedocs.io/ en/stable/. [Accessed Nov. 11, 2025].
[27] HDWallet, "Python-based library implementing a hierarchical deterministic (HD) wallet," *HDWallet documentation*, 2025. [Online]. Available: https://hdwallet.readthedocs.io/projects/python/en/master/. [Accessed Dec. 12, 2025].
[28] M. Fenniak, "PyPDF2: A free and open source pure-Python PDF library," *PyPDF2 documentation*, 2025. [Online]. Available: https://pypdf2.readthedocs.io/en/3.x/. [Accessed Nov. 11, 2025].
[29] New Relic Inc., "New Relic: Monitor, debug and improve your entire stack," New Relic, 2023. [Online]. Available: https://newrelic.com/. [Accessed Dec. 12, 2025].
[30] Etherscan, "Sepolia testnet," *Sepolia Etherscan*, 2023. [Online]. Available: https://sepolia.etherscan.io/. [Accessed Jul. 17, 2025].