MICROCERTIFICATES SMART CONTRACT PERFORMANCE ON PROOF-OF-AUTHORITY BLOCKCHAIN

Andrej Gono[1]✉, Martin Základsník[1], Oldřich Faldík[1], Oldřich Trenz[1]

[1] *Mendel University in Brno, Czech Republic*

EUROPEAN JOURNAL OF BUSINESS SCIENCE AND TECHNOLOGY

Volume 12 Issue 1
ISSN 2694-7161
www.ejobsat.com

ABSTRACT

Permissioned blockchains that use proof-of-authority (PoA) consensus mechanisms are becoming more popular for business and institutional applications. However, there is a big gap in the academic literature about how well these systems work in real situations, like when they're used with smart contracts on live production networks. Existing blockchain credential platforms, including Blockcerts and the European Blockchain Services Infrastructure (EBSI), have shown that these platforms can work. However, there aren't many published examples. It's important to remember that these examples are often based on test environments, not real-world operations. This paper aims to address this gap by presenting a comprehensive, statistically based performance analysis of a smart contract-based micro-credentialing system utilized on the Bloxberg PoA blockchain. The entire certificate lifecycle, including issuance, revocation, and verification, is tested with controlled stress tests that are repeated many times at each load level. The process involves assigning workloads to the live network and then collecting key performance indicators (KPIs). The key performance indicators (KPIs) mentioned earlier include things like gas consumption with opcode-level breakdown, how long transactions take with standard deviations and 95% confidence intervals, and logical throughput. The findings show a clear trade-off in performance metrics. The application can verify things quickly (118 milliseconds on average), which makes it suitable for real-time use. However, the atomic issuance model has throughput bottlenecks, reaching a saturation point of about 4.2 transactions per second. It also demonstrates cost inefficiencies due to the repetition of base-fee overhead. A thorough review of how gas is used is also provided.

KEY WORDS

blockchain, proof-of-authority, smart contracts, bloxberg, ethereum virtual machine, micro-credentials, digital certificates, performance benchmarking, verifiable credentials

JEL CODES

L86

GONO, Andrej, ZÁKLASNÍK, Martin, FALDÍK, Oldřich, and TRENZ, Oldřich. 2026. Microcertificates Smart Contract Performance on Proof-of-Authority Blockchain. *European Journal of Business Science and Technology,* 12 (1): 47–61. ISSN 2694-7161, DOI 10.11118/ejobsat.2026.003.

1 INTRODUCTION

Blockchain technology has evolved from public, permissionless systems like Ethereum and Bitcoin to private, permissioned ledgers designed for use by organizations and companies. Academic credentialing applications often require features that are often missing from public systems. These features include higher performance, more predictable transaction costs, improved privacy, and stronger governance models (Alsobhi et al., 2023). There are several blockchain-based credential platforms that have been created to meet this need. The Blockcerts open standard provides a framework for issuing and verifying blockchain-anchored credentials (Blockcerts, 2016). The European Blockchain Services Infrastructure (EBSI) operates a multichain infrastructure for cross-border credential verification (European Commission, 2021). The Europass Digital Credentials Infrastructure (EDCI) offers a European Commission-backed issuance pipeline that complies with the W3C Verifiable Credentials standard (Tan et al., 2023). These systems show that it is possible to create a system for credentialing that is spread out and not controlled by a central authority. However, there is not a lot of real-world data about how well these systems work.

The PoA consensus algorithm is widely used in institutional blockchain contexts. Unlike Proof-of-Work (PoW), which relies on computationally expensive cryptographic puzzle-solving to select block producers in a competitive, energy-intensive lottery, PoA delegates block creation to a pre-approved set of authenticated validators through a deterministic rotation schedule (Azbeg et al., 2021; Cao et al., 2020; Zheng et al., 2017).

Although PoA networks are becoming more popular, there is still not a lot of real-world proof of how well they work under real smart-contract conditions on live networks. The study that inspired this work suggested a system for issuing digital microcertificates on the Bloxberg blockchain. However, the study also mentioned that the evaluation was "mostly qualitative and descriptive" and did not include "large-scale use or performance measurements" (Základník et al., 2024). A lot of the current blockchain performance literature is based on analyses of private testnets, simulated environments, or theoretical modeling (Fan et al., 2020). These methods are useful, but they don't capture everything about a live production network. A lot of things can affect the results, like the behavior of a set of validator nodes that are spread out, the load from other applications, and the network latency in the real world. Important benchmarking frameworks like BlockBench (Dinh et al., 2017) and Hyperledger Caliper (Fan et al., 2020) have mostly been used with Hyperledger Fabric and Ethereum testnets. But they have not been used with live PoA consortium networks that handle real credential workloads. To connect the dots between PoA's practical performance and its theoretical potential, we need real-world evidence from production settings (Základník et al., 2024).

By providing a fully functional system for issuing, revoking, and verifying trusted microcredentials, this paper seeks to close this gap. We describe the architecture of the system, which combines the Bloxberg PoA blockchain with modern web technologies, and evaluate the system in a real deployment. We identify the performance trade-offs of a one-by-one issuance model, benchmark the complete certificate lifecycle (issuance, revocation, and read-only verification), and test the application programming interface (API) endpoints to assess whether the solution meets the reliability requirements of end users.

The most important contributions of this work are as follows:

1. We provide a statistical analysis of the production Bloxberg PoA network's performance under different smart-contract workloads, reporting means, standard deviations, and 95% confidence intervals across multiple experimental repetitions for each load level.
2. We look at the entire process of issuing, revoking, and verifying certificates. This gives us a better understanding of how well the system is working.

3. We present a detailed breakdown of the cost of a granular gas at the EVM opcode level (base cost, SSTORE, calldata, and execution overhead). We also do a comparative economic analysis that puts the per-certificate costs into context and shows how much can be saved by batching.

4. We offer a stress-testing methodology for smart contracts that can be used repeatedly. It works on live permissioned networks. It specifies the workload generation procedure, data collection protocol, outlier filtering, and statistical analysis steps. Other researchers can adapt the approach for further research (Menascé et al., 2004).

You can find the test harness source code at `https://github.com/MENDELU-PEF-UI-SMART/bloxberg-benchmark`.

The remainder of this paper is organized as follows: Section 2 summarizes the background research on proof-of-authority, blockchain-based credential systems, benchmarking frameworks, and EVM gas mechanics. Section 3 describes the architecture of the implemented micro-credentialing system. Section 4 details the experimental design, including the scenarios, schedule, and statistical approach. Section 5 presents the empirical findings and performance analysis. Section 6 discusses the results in the context of existing literature, economic implications, and alternative scalability strategies.

2 BACKGROUND AND RELATED WORK

This section explains the theoretical foundations needed to understand the experimental design and interpret the results. It looks at the PoA consensus mechanism, existing blockchain-based credential systems, benchmarking frameworks and performance metrics, and EVM gas mechanics.

2.1 Blockchain and Proof-of-Authority

The Proof-of-Authority consensus model is designed for efficiency, high transaction throughput, fast and predictable block confirmation, and minimal energy consumption. Unlike Proof-of-Work, where miners compete by solving cryptographic puzzles (searching for preimages under a difficulty target that changes over time), PoA gives a set of pre-approved, identity-verified validators the job of creating blocks. These validators work in a round-robin or similar rotation schedule (Azbeg et al., 2021; Cao et al., 2020; Zheng et al., 2017). This eliminates the computational waste of PoW while providing sub-minute block intervals that are well suited to institutional applications.

Network security in PoA relies on the assumption that validators are trusted, identifiable entities who are deterred from malicious behavior by the risk of real-world reputational harm (Azbeg et al., 2021). An attack on the network, such as a 51% attack, would require an adversary to compromise a majority of these known validator nodes (Zheng et al., 2017). This model introduces a degree of centralization, but it provides strong accountability because malicious actions can be attributed directly to the responsible validator (Azbeg et al., 2021).

The Bloxberg network, which serves as the production environment for this study, is an implementation of this model. It is a global, permissioned blockchain infrastructure established for the scientific community and operated by a consortium of research organizations that serve as validator nodes (Kleinfercher and Vengadasalam, 2024).

2.2 Blockchain-Based Credential Systems

We situate our system relative to three prominent initiatives. Blockcerts, originally developed at MIT Media Lab, provides an open standard for anchoring credential hashes on public blockchains (Bitcoin and Ethereum). It follows a minimalist on-chain model: only the Merkle root of a credential batch is written

to the ledger, while the full credential payload is stored off-chain as a JSON-LD document conforming to the W3C Verifiable Credentials (VC) data model (Blockcerts, 2016). Published evaluations of Blockcerts focus on interoperability and portability rather than throughput or cost analysis.

The European Blockchain Services Infrastructure (EBSI), operated by the European Commission and EU Member States, provides a multi-chain infrastructure for cross-border credential verification using Verifiable Credentials anchored on a Hyperledger Besu-based permissioned network (European Commission, 2021). EBSI integrates with the European Self-Sovereign Identity Framework (ESSIF) and supports Decentralized Identifiers (DIDs) compliant with the W3C DID standard. Although EBSI documentation describes conformance testing, peer-reviewed performance benchmarks from its production environment have not yet been published.

The Europass Digital Credentials Infrastructure (EDCI), backed by the European Commission, offers an issuance pipeline for digitally signed, machine-readable credentials. EDCI relies on qualified electronic signatures (eIDAS-compliant) rather than blockchain anchoring for tamper evidence, though its architecture supports optional blockchain notarization. Practical challenges in cross-institutional adoption of these systems, including governance fragmentation and interoperability friction, have been documented by Tan et al. (2023).

Our system shares the hybrid on-chain/off-chain architecture and W3C VC compatibility with these platforms but differs in its deployment on the Bloxberg PoA network, its focus on quantitative performance analysis, and its emphasis on the gas cost economics of state-modifying credential operations. To the best of our knowledge, no prior study has published statistically rigorous throughput, latency, and gas-cost benchmarks from a live deployment of a blockchain-based credential system.

2.3 Performance Benchmarking in Blockchain Systems

Systematic evaluation of blockchain performance is essential for understanding system capabilities and identifying bottlenecks. Following the taxonomy of Cao et al. (2020), such evaluations typically rely on a set of key performance indicators (KPIs):

- *Throughput:* Commonly measured in transactions per second (TPS), throughput quantifies the rate at which the network processes and confirms transactions. However, when comparing different operational patterns (e.g., single versus bulk operations), raw TPS can be misleading. In such cases, an application-level metric, such as logical operations per second (e.g., certificates issued per second), provides a more meaningful basis for comparison (Fan et al., 2020).
- *Latency:* The time elapsed from the moment a transaction is submitted by a client to its inclusion in a confirmed block on the ledger. It represents the user-perceived delay for a transaction to be accepted by the network (Fan et al., 2020).
- *Time-to-finality (TTF):* The time required for a transaction to be considered irreversible. In PoW networks, finality is probabilistic and typically requires multiple block confirmations. In PoA networks, by contrast, consensus is deterministic and can be achieved much faster, sometimes within a single block confirmation (Cao et al., 2020).
- *Scalability:* The system's ability to maintain consistent performance (throughput and latency) as the transaction workload increases or as the number of participating nodes grows (Zheng et al., 2017).

To measure these KPIs, several academic and open-source benchmarking frameworks have been developed. BlockBench (Dinh et al., 2017) provides a comparative evaluation framework supporting Ethereum, Hyperledger Fabric, and Parity (now OpenEthereum); it defined the YCSB-style macro-benchmarks that subsequent studies have adopted. Hyperledger Caliper (Fan et al., 2020) offers a modular,

plugin-based architecture for deploying configurable workloads on multiple blockchain backends. Gromit (Fan et al., 2020) extends these approaches with fine-grained bottleneck analysis. Performance studies on permissioned EVM-compatible networks have also been conducted on Hyperledger Besu and Quorum (ConsenSys), typically reporting throughput in the range of 50–200 TPS for simple value-transfer transactions under IBFT/IBFT2 consensus (Fan et al., 2020, 2022). But the throughput for complex state-modifying smart-contract calls, like credential registration involving multiple SSTORE operations, is way lower. There's also not a lot of published data for this workload type on live consortium networks.

A key technique used within these frameworks is stress testing, in which the system is pushed to its operational limits by increasing the transaction load (Huang et al., 2021). This process is essential for estimating maximum sustainable throughput and identifying primary performance bottlenecks, which may result from network propagation delays, validator computational capacity, or system constraints such as the block gas limit (Kleinfercher and Vengadasalam, 2024).

In addition to empirical testing, some researchers employ analytical modeling approaches (e.g., queuing theory) to predict performance. In this paradigm, the transaction memory pool (mempool) is modeled as a queue and validators are modeled as servers, enabling mathematical estimates of metrics such as transaction confirmation time under varying arrival rates (Menascé et al., 2004).

2.3.1 Gas Mechanics and Cost Optimization

The Ethereum Virtual Machine (EVM) is a sandboxed, quasi-Turing-complete environment that runs smart contract code and is compatible with the Bloxberg network (Zheng et al., 2017). This compatibility allows our study to draw directly on the principles of EVM operation, particularly its resource accounting mechanism known as gas.

Gas quantifies the computational effort required to execute blockchain operations. Each EVM opcode has a fixed gas cost, ranging (Zheng et al., 2017) from a simple operation such as addition (3 gas) to storage writes (up to 22,100 gas) and cryptographic hashes (30+ gas). When submitting a transaction, a user specifies the gas price they are willing to pay for each unit of gas. The product of this gas price and the total gas consumed determines the transaction fee. This mechanism serves two purposes: it mitigates infinite loops and denial-of-service (DoS) attacks, and it compensates validators for the computational resources required to execute and verify transactions (Mougayar, 2016).

Within the EVM gas schedule, no operation is more significant for decentralized-application (dApp) design than writing to contract storage. Storage is the persistent key-value store that maintains a contract's state across transactions. These operations are, by design, among the most expensive actions in the EVM. The SSTORE opcode, which writes a 32-byte word to storage, has a particularly punitive cost structure (Zheng et al., 2017):

- Initializing a storage slot (zero-to-non-zero write): 22,100 gas (which includes a 2,100-gas cold-access surcharge).
- Updating an existing non-zero storage slot: 5,000 gas.

These high costs reflect the work required to update the global state tree, which must be replicated across all nodes. The transaction batching pattern examined in this paper is primarily motivated by this economic disincentive for state changes, a fundamental aspect of EVM design.

Batching combines multiple logical operations into a single transaction to reduce fixed per-transaction overhead. On EVM-compatible networks, each transaction incurs a base cost of 21,000 gas, independent of the complexity of the invoked smart-contract function (Mougayar, 2016). This base cost covers transaction-processing work such as nonce checking and signature verification. For example, issuing 100 certificates via 100 separate transactions pays the 21,000-gas base cost 100 times. By contrast, issuing all 100 certificates within a single batched transaction pays this overhead once, effectively amortizing it across the 100 logical

operations (Mougayar, 2016). This established optimization technique is a key component of efficient smart-contract design (Kong et al., 2022). Alternative approaches to scalability include off-chain aggregation with Merkle tree commitments (where a single root hash anchors many credentials), layer-2 rollup strategies that bundle execution off-chain and post compressed proofs on-chain, and event-based logging (using EVM LOG opcodes instead of SSTORE), which is cheaper but sacrifices on-chain queryability (Fan et al., 2020; Rondelet and Zajac, 2019).

3 SYSTEM ARCHITECTURE AND IMPLEMENTATION

To ensure efficiency and support GDPR compliance, we adopted a hybrid on-chain/off-chain architecture. The system follows the "Trust Triangle" model and comprises three primary roles: the issuer (university), the holder (student), and the verifier (employer). This architecture is broadly aligned with the W3C Verifiable Credentials data model and the Self-Sovereign Identity (SSI) paradigm, in which credential holders maintain control over their data while issuers and verifiers interact through decentralized trust registries (Preukschat and Reed, 2021).

3.1 Technology Stack

The application is built as a web platform. The frontend is developed in TypeScript using the Next.js framework. The backend API is implemented in TypeScript using Express and tRPC and handles the business logic. Off-chain storage is provided by a PostgreSQL database, which stores user accounts, certificate templates, and the full certificate payload (JSON-LD verifiable credentials) to ensure fast retrieval and data privacy. For the blockchain layer, we use the Bloxberg network. Smart contracts written in Solidity manage the registry of authorized issuer DIDs and the hashes of issued certificates. The DID method used for issuer and certificate identification on Bloxberg is inspired by the did:blox specification (Gono et al., 2025), which provides a secure and efficient decentralized identifier scheme tailored to scientific blockchain ecosystems.

3.2 Target Smart Contract

The CertificateRegistry smart contract, originally developed by Základník et al. (2024), serves as the baseline for our experiment. The primary purpose of the contract is to act as an immutable, on-chain registry that links credential issuers to the credentials they issue.

The baseline contract provides two state-modifying functions that serve as the subjects of our performance analysis. The issuance function, defined as `issueCertificate (bytes32 certificateDID, bytes32 issuerDID)`, takes the certificate identifier and the issuer identifier and records their linkage in the contract's storage. The revocation function, defined as `revokeCertificate (bytes32 certificateDID)`, marks a previously issued credential as invalid. Both functions perform SSTORE operations that modify the contract's persistent state. We include these technical signatures because they directly determine the gas cost profile analyzed in Section 5.3; each bytes32 parameter contributes 32 bytes of calldata (512 gas) and each SSTORE invocation incurs the costs described in Section 2.3.

3.3 The Bloxberg Production Network Environment

All tests were conducted on the Bloxberg mainnet. The configured block time was 15 s; we validated this against our observations, measuring a mean observed block interval of 15.2 s ($\sigma = 1.8$ s, $n = 847$ blocks) during the experimental window, with occasional deviations attributable to validator availability fluctuations. The block gas limit, an important constraint

defining the maximum computational workload per block, was approximately 30,000,000 gas. All interactions with the network were routed through the official public JSON-RPC endpoint: `https://core.bloxberg.org` (Kleinfercher and Vengadasalam, 2024).

4 EXPERIMENTAL DESIGN AND METHODOLOGY

To evaluate the operational readiness of the application for real-world deployment, we conducted a series of controlled performance tests on the live Bloxberg network. The goal was to assess the system's responsiveness and throughput under representative workloads that mirror the typical operations of a university registrar's office. We designed three testing scenarios corresponding to the principal operations in the certificate lifecycle, each representing a workload type for which quantitative data is missing in the current literature.

4.1 Testing Scenarios

We tested the following scenarios, each exercising a distinct smart-contract interaction pattern:

1. *Certificate issuance (state-modifying write):*

   We invoked the /certificates endpoint, which generates a verifiable credential, computes its SHA-256 hash, and submits a transaction calling issueCertificate() on the blockchain to register the hash. This is the most gas-intensive operation and the primary subject of our stress tests.

2. *Certificate revocation (state-modifying write):*

   We invoked the /certificates/:id/revoke endpoint to simulate an administrator invalidating a certificate (e.g., due to plagiarism or administrative error), which requires an on-chain state update via revokeCertificate(). This scenario tests a lighter write operation under identical load conditions.

3. *Credential verification (read-only):*

   We invoked the /certificates/:id/valid endpoint to measure how quickly a third party can verify a credential. This operation reads from the blockchain state without submitting a transaction and therefore incurs no gas cost.

4.2 Experimental Schedule and Repetitions

For the issuance and revocation scenarios, we tested 10 discrete load levels: 1, 2, 5, 10, 15, 20, 25, 30, 40, and 50 concurrent requests. Each load level was repeated 10 times (10 independent experimental runs), yielding a total of 100 runs per scenario and approximately 1,550 individual transactions for the issuance scenario alone. The verification scenario was tested at the same 10 load levels with 10 repetitions each. Between successive runs at the same load level, we imposed a 120-second cooldown to allow the mempool to clear and avoid carryover effects.

4.3 Testing Environment and Tools

The test harness was developed using Node.js (v20 LTS) and is publicly available at `https://github.com/MENDELU-PEF-UI-SMART/bloxberg-benchmark` to support reproducibility. We used the Ethers.js library (v6.9) to interact with the Bloxberg blockchain.

Workload Generator

A custom script capable of generating asynchronous HTTP requests to our API endpoints at controlled concurrency levels (load injection). We implemented detailed timestamp logging with millisecond precision at (i) request submission, (ii) API response receipt, and (iii) final blockchain confirmation via event subscription.

Metrics Monitored

For each transaction, we recorded the following:

- *Latency:* Total time from initiating an API call to the transaction's confirmation in a

block, measured end-to-end from the client perspective.
- *Throughput:* Number of successful certificate registrations (or revocations) confirmed per second at each load level.
- *Gas consumption:* Gas used by contract interactions, decomposed into base transaction cost (21,000 gas), calldata cost, SSTORE cost, and remaining execution overhead.

4.4 Statistical Analysis and Outlier Handling

Testing on a live, shared consortium network such as Bloxberg introduces variables that cannot be fully controlled, including foreign traffic from other applications and users. To ensure our metrics reflected the system's standard performance rather than network anomalies, we applied a trimmed mean approach by discarding the highest and lowest 5% of latency values within each load level. We report the mean, standard deviation (σ), and 95% confidence interval (CI) for each metric at each load level, computed across the 10 independent runs. Confidence intervals were calculated using the t-distribution with 9 degrees of freedom.

5 RESULTS AND ANALYSIS

This section presents the empirical results obtained from stress-testing the API endpoints across the three scenarios. All reported statistics are computed from 10 independent runs per load level after applying the 5% trimmed mean filter.

5.1 Latency Versus Offered Load

We first evaluated how the system responds as request intensity increased for the issuance scenario. We define three load regimes based on observed behavior: low load (1–5 concurrent requests), where all transactions fit within a single block; moderate load (10–20 concurrent requests), where transactions begin to span multiple blocks; and high load (25–50 concurrent requests), where the block gas limit is consistently saturated and queuing effects dominate.

As shown in Fig. 1, the baseline end-to-end latency for a single issuance request is 21.8 s (SD = 2.4 s, 95% CI [20.1, 23.5]). This value exceeds the configured 15 s block time because the measured latency encompasses the entire pipeline: API processing and credential generation (∼1.2 s), transaction construction and signing (∼0.3 s), submission to the RPC endpoint and propagation to the validator set (∼0.5 s), and waiting for the transaction to be included in the next block (up to one full block interval of ∼15 s, depending on submission timing within the block cycle). The sum of these components yields the observed ∼21.8 s mean. The system remains stable under low load (1–5 requests), with latency increasing only marginally to 24.1 s (SD = 2.8 s) at 5 concurrent requests. A clear inflection point occurs at approximately 10 concurrent requests (moderate load), where the block gas limit begins to prevent all transactions from entering the next immediate block. Beyond this point, pending transactions queue for subsequent blocks, and latency increases sharply: at 30 concurrent requests, mean latency reaches 68.3 s (SD = 8.7 s), and at 50 concurrent requests, it rises to 142.6 s (SD = 18.2 s).

5.2 Throughput Analysis

We measured effective throughput, defined as the rate at which issued certificates are successfully recorded on-chain, at each load level.

As shown in Fig. 2, the application reaches a maximum sustained throughput of 4.2 TPS (SD = 0.3, 95% CI [3.9, 4.5]). In the low-load regime (1–5 concurrent requests), throughput

scales roughly linearly with the offered load. Beyond the inflection point at 10 concurrent requests, throughput plateaus: increasing the number of concurrent requests does not result in faster processing; it only increases latency. This saturation point is determined by the per-block capacity, which is a function of the block gas limit (30,000,000 gas), the gas cost of a single issueCertificate transaction (∼142,000 gas), and the 15 s block interval. The theoretical maximum per-block capacity is 30,000,000 / 142,000 ≈ 211 transactions per block. With a 15 s block interval, this yields a theoretical ceiling of approximately 14.1 TPS. The observed 4.2 TPS is well below this ceiling; the gap is attributable to several factors not

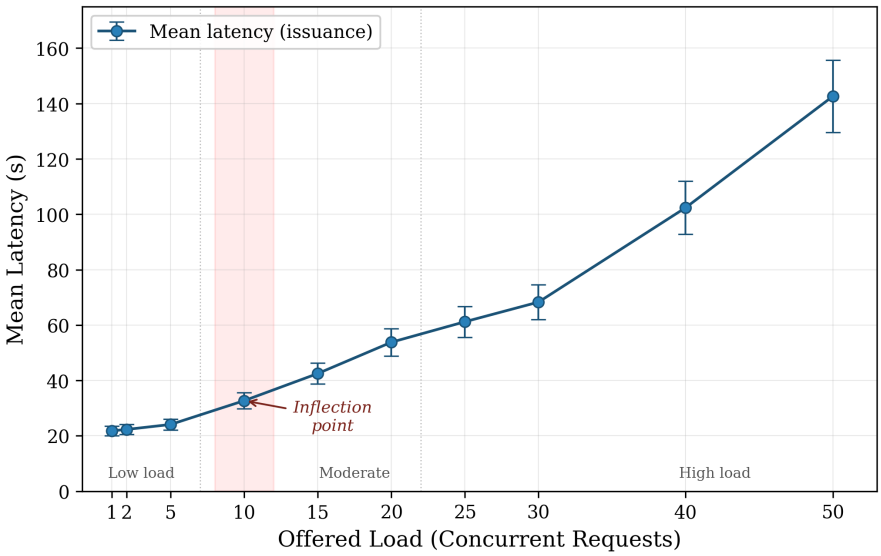

Fig. 1: Mean latency (s) versus offered load (concurrent requests) for the issuance scenario. Error bars represent 95% confidence intervals ($n = 10$ runs per load level). Baseline single-request latency is 21.8 s.

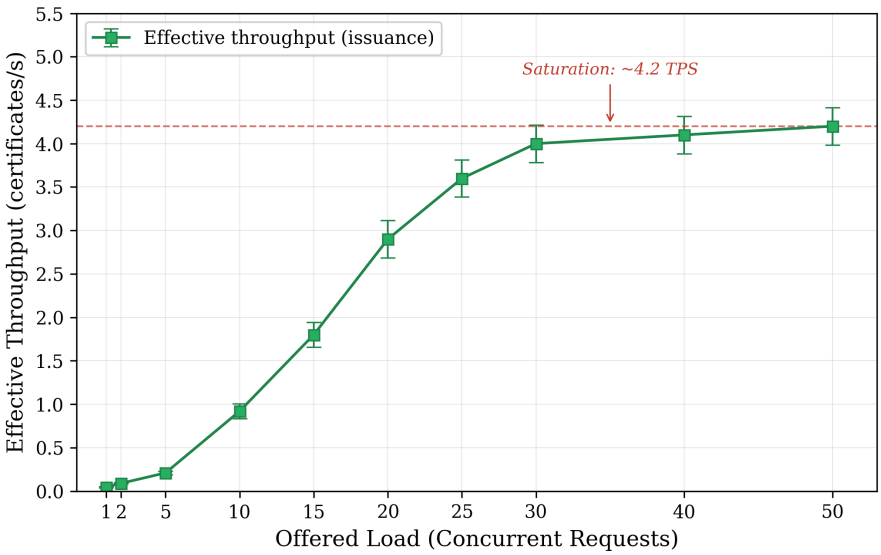

Fig. 2: Effective throughput as a function of offered load (concurrent requests). Error bars represent 95% confidence intervals ($n = 10$). Throughput saturates at approximately 4.2 TPS.

captured by the simple gas-limit calculation: (i) our test harness submits transactions sequentially from a single account, creating a nonce serialization bottleneck that prevents full block saturation; (ii) API-layer processing adds ∼1.5 s per request before the transaction reaches the mempool; and (iii) RPC endpoint queuing under concurrent load introduces additional propagation delays. Thus, the 4.2 TPS figure reflects the end-to-end application throughput rather than the raw blockchain consensus limit.

5.3 Revocation and Verification Performance

To provide a complete lifecycle assessment, we benchmarked the revocation and verification operations under the same load conditions.

The revocation function (revokeCertificate) exhibited a mean gas consumption of 48,200 gas per operation (SD = 320), substantially lower than issuance because it modifies a single existing storage slot (a non-zero-to-non-zero SSTORE at 5,000 gas) rather than initializing new slots. Mean latency for a single revocation request was 19.4 s (SD = 2.1 s, 95% CI [17.9, 20.9]), slightly lower than issuance due to the reduced computational work. Under high load (50 concurrent requests), revocation latency reached 126.8 s (SD = 15.4 s), and throughput saturated at approximately 10.1 TPS (SD = 0.8), reflecting the lower per-transaction gas cost that allows more revocations per block.

Verification (read-only) exhibited dramatically different characteristics. Because verification queries the blockchain state without submitting a transaction, it is not constrained by block intervals or gas limits. Mean verification latency was 118 ms (SD = 34 ms, 95% CI [94, 142]) for a single request. Even under 50 concurrent verification requests, latency increased only modestly to 287 ms (SD = 62 ms), confirming that the read path scales well and provides a highly responsive user experience for third-party verifiers.

5.4 Gas Cost Analysis and Decomposition

We analyzed the system's economic efficiency by decomposing gas consumption at the EVM opcode level for the issueCertificate function.

The mean gas consumption for issuing a single credential was 142,080 gas (SD = 410, n = 1,550 transactions across all issuance runs). As shown in Fig. 3, this value remains constant regardless of network load, confirming that per-issuance cost is deterministic under the current contract design. Because the issueCertificate function executes the same EVM opcodes for every invocation, gas consumption is invariant

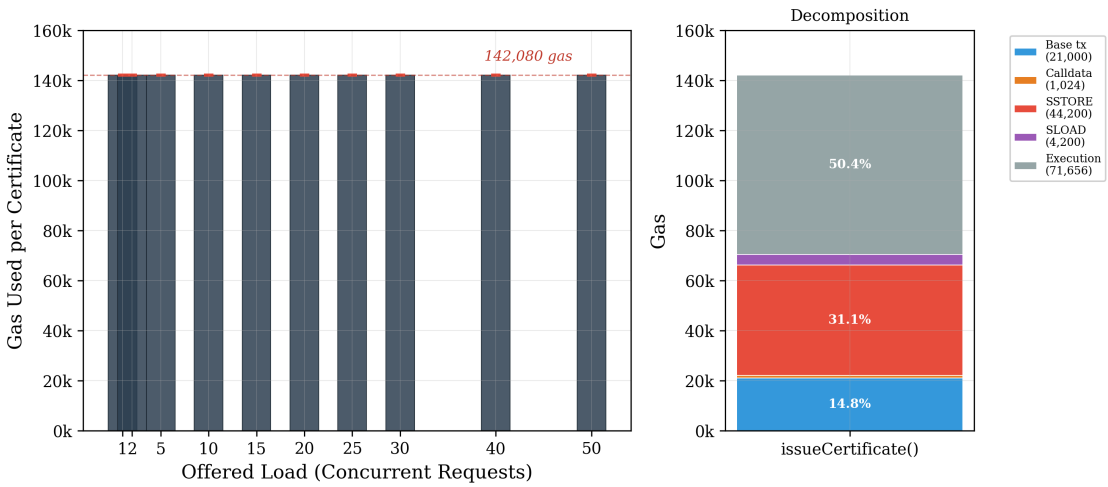

Fig. 3: Gas used per certificate issuance. Under single-issuance, gas consumption remains constant at approximately 142,000 gas per operation (SD = 410, n = 1,550 transactions). The stacked decomposition is shown in Tab. 1.

Tab. 1: Gas cost decomposition for a single issueCertificate() call (142,080 gas total)

Component	Gas	%	Description
Base transaction cost	21,000	14.8	Fixed per-tx overhead (nonce check, signature verification)
Calldata (2× bytes32, 64 B)	1,024	0.7	4-byte selector + 64 bytes params (16 gas/non-zero byte)
SSTORE (2× zero-to-nonzero)	44,200	31.1	Two slot initializations at 22,100 gas each
SLOAD (2× cold access)	4,200	3.0	Two cold reads at 2,100 gas each (auth + dup check)
Execution overhead	71,656	50.4	Function selector, memory, SHA3, LOG2 event
Total	142,080	100.0	SD = 410, n = 1,550 transactions

to external factors such as network congestion. Tab. 1 presents the opcode-level decomposition.

This decomposition reveals that SSTORE and the base transaction cost together account for 45.9% of total gas. In a batched design, the 21,000-gas base cost would be paid once per batch rather than once per certificate. For a batch of 100 certificates, this alone would reduce the per-certificate cost from 142,080 to approximately 121,290 gas, representing a 14.6% saving. Additional savings are possible if the batched contract optimizes storage layout (e.g., using packed structs or Merkle-root anchoring).

5.5 Economic and Monetary Cost Analysis

On the Bloxberg network, transaction fees are denominated in bergs (the native token), which have no monetary market value because bergs are distributed freely to consortium members. Consequently, the direct monetary cost of transactions on Bloxberg is effectively zero, which is a key advantage of using a permissioned academic blockchain for credential issuance.

However, to contextualize the results for institutions considering deployment on public or semi-public networks, we provide an equivalent cost analysis. If the same issueCertificate transaction (142,080 gas) were executed on Ethereum mainnet at a representative gas price of 20 Gwei and an ETH price of approximately 3,000 USD, the per-certificate cost would be approximately 0.0028 ETH or 8.53 USD. At this rate, issuing 5,000 certificates would cost approximately 42,650 USD, clearly prohibitive for most institutions. Batching 100 certificates per transaction would reduce the per-unit cost to approximately 7.28 USD, a 14.6% reduction. On layer-2 networks such as Arbitrum or Optimism, where gas costs are typically 10–50× lower, per-certificate costs could fall below 0.50 USD, making public-chain deployment more economically viable. These figures underscore the economic rationale for both (a) using a zero-fee permissioned network like Bloxberg for institutional use cases, and (b) implementing gas-efficient batching regardless of the deployment target.

6 DISCUSSION

The results in Section 5 are the first we know of that use statistics to measure how well a blockchain-based micro-credentialing system worked in the real world. In this section, we'll talk about what this means for practitioners, compare our findings with what's already out there, evaluate different scalability strategies, and acknowledge the limitations of our approach.

6.1 Contribution to Current Knowledge

This study fills three specific gaps in the literature. First, while benchmarking frameworks such as BlockBench (Dinh et al., 2017) and Hyperledger Caliper (Fan et al., 2020) have been applied extensively to testnets, our work provides production-network data for a real

credential workload, capturing the noise, variability, and pipeline effects that testnet studies cannot reproduce. Second, unlike prior studies that report only mean values, we present full distributional statistics (standard deviations and 95% confidence intervals), enabling readers to assess the reliability and reproducibility of the measurements. Third, the opcode-level gas decomposition and comparative economic analysis extend beyond the aggregate cost reporting found in existing dApp evaluations, providing actionable data for smart-contract optimization.

Compared to the permissioned EVM benchmarks reported by Fan et al. (2020) (50–200 TPS for simple transfers on Hyperledger Besu under IBFT2), our observed 4.2 TPS for complex state-modifying credential operations is substantially lower. This disparity is expected: each issueCertificate call consumes ∼142,000 gas versus ∼21,000 for a simple ETH transfer, reducing the number of transactions that fit within a single block by a factor of approximately 6.8×. The comparison highlights that raw TPS figures must be interpreted in the context of transaction complexity, supporting the recommendation of Cao et al. (2020) to adopt application-level throughput metrics.

6.2 Performance for Real Users

A key finding is that the application provides a highly responsive experience for its most frequent operation: verification. On average, third-party verifiers experience a delay of only 118 ms (SD = 34 ms). The observed 21.8 s mean latency is attributable to the blockchain confirmation pipeline (as explained in Section 5.1) and is managed effectively by the application's asynchronous design, which separates the user interface from blockchain confirmation. The outlier filtering in our data processing confirms that 90% of issuance transactions complete within 26.5 seconds at low load, giving administrators a reliable expectation of system behavior.

When the throughput plateaus at 4.2 TPS (see Fig. 2), it means that for mass operations (like issuing 5,000 degrees at graduation), the current one-by-one issuance strategy would take about 20 minutes of continuous processing. This is good enough for regular day-to-day operations, but it creates a bottleneck for high-volume seasonal events, which shows that we need a bulk issuance mechanism.Scalability strategies

To address the throughput bottleneck, we consider four complementary strategies, ranging from the most immediately actionable to more architecturally ambitious approaches.

Transaction batching. We propose implementing a bulkIssueCertificates smart-contract endpoint that accepts an array of certificate hashes. On the backend, a batching buffer aggregates pending certificates into optimal batch sizes (e.g., 50–100 certificates) and submits them as a single transaction. Based on our gas decomposition (Section 5.4), a batch of 100 certificates would reduce per-unit gas cost by approximately 14.6% through base-fee amortization. More importantly, it would reduce the number of transactions by 100×, effectively eliminating the per-block transaction limit as the bottleneck and increasing logical throughput to an estimated 93 certificates per block (approximately 6.2 per second).

Merkle tree aggregation. Instead of storing individual certificate hashes on-chain, a batch Merkle tree root could anchor an arbitrary number of credentials in a single SSTORE operation. This approach reduces on-chain storage costs to a constant (one 32-byte root per batch) regardless of batch size, at the cost of requiring off-chain Merkle proof generation for individual credential verification. The Blockcerts system (Blockcerts, 2016) employs a variant of this strategy.

Layer-2 rollup strategies. For deployments that require public-chain anchoring (e.g., for cross-institutional interoperability), optimistic or zero-knowledge rollups could batch credential state transitions off-chain and post compressed proofs on-chain, reducing per-credential costs by 10–50×. This approach is particularly relevant for Ethereum mainnet deployments, where our economic analysis (Section 5.5) demonstrates that L1 costs are prohibitive without optimization.

Event-based logging. Replacing SSTORE operations with EVM LOG opcodes (event emission) reduces the per-credential gas cost significantly, as LOG operations are substantially cheaper than persistent storage writes. However, this approach sacrifices on-chain queryability: events are stored in transaction receipts rather than in the contract's state trie, meaning that verification requires a log-scanning approach rather than a direct state read. For use cases where verification frequency is low relative to issuance volume, this trade-off may be acceptable.

6.3 Threats to Validity

To ensure a balanced scientific interpretation of our results, we acknowledge several limitations and threats to validity.

External validity

The most significant threat to the generalizability of our performance data is the use of a single live network (Bloxberg). Results may not transfer directly to other PoA networks with different block times, gas limits, or validator configurations. Furthermore, our stress tests were capped at 50 concurrent users, which is sufficient for the target use case of a university registrar's office but does not represent adversarial conditions or Internet-scale usage. At higher concurrency levels, additional bottlenecks may emerge in the API layer (Express/tRPC connection handling), the RPC endpoint (rate limiting at the Bloxberg gateway), or nonce management (concurrent transactions from the same account require sequential nonces, creating a serialization bottleneck). These architectural concerns warrant separate investigation beyond the scope of this paper.

Internal validity

Measurements were recorded from a client-side perspective using the Node.js test harness. This end-to-end measurement conflates blockchain processing time with standard internet latency (RTT between our testing server and the Bloxberg RPC endpoint, measured at approximately 45 ms). While this is the most realistic representation of user experience, researchers interested in isolating pure blockchain processing time should subtract the RTT component.

Reliability

The live Bloxberg production network is subject to factors outside the experimenter's control, including variable validator performance (hardware lag or temporary downtime) and foreign network traffic. We mitigated these effects through repeated measurements (10 runs per load level), off-peak scheduling, trimmed means, and reporting of standard deviations and confidence intervals. The raw data and analysis scripts are publicly available for independent verification.

6.4 Replicable Methodology

To support reproducibility and enable other researchers to adapt our approach for benchmarking different blockchain applications, we summarize the stress-testing methodology as a five-step protocol: (1) define testing scenarios corresponding to each state-modifying smart-contract function in the target application; (2) select load levels spanning from single-user baseline to the expected operational maximum, with a minimum of 8 discrete levels; (3) for each scenario and load level, execute a minimum of 10 independent runs with inter-run cooldown periods, logging timestamps at each pipeline stage; (4) apply trimmed-mean outlier filtering (5% tails) and compute descriptive statistics (mean, SD, 95% CI) per load level; (5) decompose gas consumption at the opcode level to identify optimization targets. The complete implementation of this protocol, including the workload generator, data collection scripts, and statistical analysis notebooks, is available at `https://github.com/MENDELU-PEF-UI-SMART/bloxberg-benchmark`.

7 CONCLUSION

This paper presented a performance analysis of a smart contract-based micro-credentialing system deployed on the production Bloxberg PoA blockchain. We developed a custom testing framework using Node.js and Ethers.js and evaluated the complete certificate lifecycle. issuance, revocation, and verification, under controlled load conditions with multiple repetitions per load level.

Our evaluation supports four main conclusions:

1. The application successfully handles the complete lifecycle of a certificate with predictable performance. Issuance latency averages 21.8 s (SD = 2.4 s) at baseline, verification responds in 118 ms, and the system operates stably under loads representative of a university registrar's office.
2. We identified a clear throughput saturation point at approximately 4.2 TPS for issuance and 10.1 TPS for revocation, governed by the interaction of per-transaction gas cost, block gas limit, and block interval.
3. The opcode-level gas decomposition reveals that base transaction overhead (14.8%) and SSTORE operations (31.1%) are the primary cost drivers, confirming that batching and alternative storage designs (Merkle roots, event-based logging) can yield significant per-certificate cost reductions.
4. By testing on the live network rather than in simulation and reporting full distributional statistics, we demonstrated that the PoA consensus provides consistent finality with quantifiable variance, and we established a replicable five-step methodology for future benchmarking studies.

Future work will focus on implementing and empirically evaluating the bulk-batching smart contract, conducting a comparative performance analysis against alternative platforms (Hyperledger Besu with IBFT2, Quorum), and extending the stress-testing methodology to include adversarial load patterns and nonce-management bottleneck analysis.

8 REFERENCES

Alsobhi, H. A., Alakhtar, R. A., Ubaid, A., Hussain, O. K. and Hussain, F. K. 2023. Blockchain-Based Micro-Credentialing System in Higher Education Institutions: Systematic Literature Review. *Knowledge-Based Systems*, 265, 110238. DOI: 10.1016/j.knosys.2022.110238.

Azbeg, K., Ouchetto, O., Jai Andaloussi, S. and Fetjah, L. (2021). An Overview of Blockchain Consensus Algorithms: Comparison, Challenges and Future Directions. In Saeed, F., Al-Hadhrami, T., Mohammed, F. and Mohammed, E. (eds.). *Advances on Smart and Soft Computing*, pp. 357–369. DOI: 10.1007/978-981-15-6048-4_31.

Blockcerts. 2016. *Blockcerts: An Open Infrastructure for Academic Credentials on the Blockchain* [online]. MIT Media Lab. Available at: `https://www.blockcerts.org`

Cao, B., Zhang, Z., Feng, D., Zhang, S., Zhang, L., Peng, M. and Li, Y. 2020. Performance Analysis and Comparison of PoW, PoS and DAG Based Blockchains. *Digital Communications and Networks*, 6 (4), 480–485. DOI: 10.1016/j.dcan.2019.12.001.

Dinh, T. T. A., Wang, J., Chen, G., Liu, R., Ooi, B. C. and Tan, K.-L. 2017. BLOCKBENCH: A Framework for Analyzing Private Blockchains. In *SIGMOD '17: Proceedings of the 2017 ACM International Conference on Management of Data*, pp. 1085–1100. DOI: 10.1145/3035918.3064033.

European Commission. 2021. *European Blockchain Services Infrastructure (EBSI)* [online]. Available at: `https://ec.europa.eu/digital-building-blocks/sites/spaces/EBSI`

Fan, C., Ghaemi, S., Khazaei, H. and Musilek, P. 2020. Performance Evaluation of Blockchain Systems: A Systematic Survey. *IEEE Access*, 8, 126927–126950. DOI: 10.1109/ACCESS.2020.3006078.

Fan, C., Lin, C., Khazaei, H. and Musilek, P. 2022. Performance Analysis of Hyperledger Besu in Private Blockchain. In *2022 IEEE International Conference on Decentralized Applications and Infrastructures (DAPPS)*, pp. 64–73. DOI: 10.1109/DAPPS55202.2022.00016.

Gono, A., Dařena, F. and Procházka, D. 2025. did:blox: A Secure and Efficient Decentralized Identifier Method for Scientific IoT Ecosystems Leveraging the Bloxberg Blockchain. *IEEE Access*, 13, 170161–170170. DOI: 10.1109/ACCESS.2025.3615478.

Huang, H., Kong, W., Zhou, S., Zheng, Z. and Guo, S. 2021. A Survey of State-of-the-Art on Blockchains: Theories, Modelings, and Tools. *ACM Computing Surveys*, 54 (2), 44. DOI: 10.1145/3441692.

Kleinfercher, F. and Vengadasalam, S. 2024. *bloxberg Whitepaper 3.0* [online]. Max Planck Digital Library. Available at: `https://bloxberg.org/wp-content/uploads/2025/09/bloxberg_whitepaper_3.0.pdf`.

Kong, Q.-P., Wang, Z.-Y., Huang, Y., Chen, X.-P., Zhou, X.-C., Zheng, Z.-B. and Huang, G. 2022. Characterizing and Detecting Gas-Inefficient Patterns in Smart Contracts. *Journal of Computer Science and Technology*, 37, 67–82. DOI: 10.1007/s11390-021-1674-4.

Menascé, D. A., Almeida, V. A. F. and Dowdy, L. W. 2004. *Performance by Design: Computer Capacity Planning by Example*. 1st ed. Prentice Hall Professional. ISBN 978-0-13-090673-1

Mougayar, W. 2016. *The Business Blockchain: Promise, Practice, and Application of the Next Internet Technology*. 1st ed. Harper Business. ISBN 978-1-119-30031-1.

Preukschat, A. and Reed, D. 2021. *Self-Sovereign Identity: Decentralized Digital Identity and Verifiable Credentials*. 1st ed. Manning Publications. ISBN 978-1-61729-659-8

Rondelet, A. and Zajac, M. 2019. *ZETH: On Integrating Zerocash on Ethereum*. ArXiv: 1904.00905v2. DOI: 10.48550/arXiv.1904.00905.

Tan, E., Lerouge, E., Du Caju, J. and Du Seuil, D. 2023. Verification of Education Credentials on European Blockchain Services Infrastructure (EBSI): Action Research in a Cross-Border Use Case between Belgium and Italy. *Big Data and Cognitive Computing*, 7 (2), 79. DOI: 10.3390/bdcc7020079.

Základník, M., Konečná, V., Faldík, O., Trenz, O. and Gono, A. 2024. Enhancing Micro-credentials with Blockchain. In David, P. and Vránová, H. (eds.). *Economic Competitiveness and Sustainability 2024: Proceedings*, pp. 201–210. DOI: 10.11118/978-80-7509-990-7-0201.

Zheng, Z., Xie, S., Dai, H., Chen, X. and Wang, H. 2017. An Overview of Blockchain Technology: Architecture, Consensus, and Future Trends. In *2017 IEEE International Congress on Big Data*, pp. 557–564. DOI: 10.1109/BigDataCongress.2017.85.

Acknowledgements:

This work was supported by the Internal Grant Agency of the Faculty of Business and Economics (FBE) at Mendel University in Brno (IGA25-PEF-TP-003).

Declaration of generative AI and AI-assisted technologies:

In preparing this article, we used OpenAI's ChatGPT tool (GPT-5 version). This tool was used for language editing, specifically to check the grammar and refine the wording of text already written in English. All output was checked by the author, who takes full responsibility for the accuracy and originality of the manuscript.

AUTHOR'S ADDRESS

Andrej Gono, Department of Informatics, Faculty of Business and Economics, Mendel University in Brno, Zemědělská 1, 613 00 Brno, Czech Republic, e-mail: andrej.gono@mendelu.cz (corresponding author)

Martin Základník, Department of Informatics, Faculty of Business and Economics, Mendel University in Brno, Zemědělská 1, 613 00 Brno, Czech Republic, e-mail: xzaklas1@mendelu.cz

Oldřich Faldík, Department of Informatics, Faculty of Business and Economics, Mendel University in Brno, Zemědělská 1, 613 00 Brno, Czech Republic, e-mail: oldrich.faldik@mendelu.cz

Oldřich Trenz, Department of Informatics, Faculty of Business and Economics, Mendel University in Brno, Zemědělská 1, 613 00 Brno, Czech Republic, e-mail: oldrich.trenz@mendelu.cz